(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 255 179 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
06.11.2002 Bulletin 2002/45

(51) Int Cl.⁷: G06F 1/00

(21) Application number: 02008495.0

(22) Date of filing: 15.04.2002

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 30.04.2001 US 846175

(71) Applicant: MICROSOFT CORPORATION
Redmond, Washington 98052-6399 (US)

(72) Inventors:
- Brezak, John E.
  Woodinville, WA 98072 (US)
- Brundrett, Peter T.
  Seattle, WA 98115 (US)
- Ward, Richard B.
  Redmond, WA 98053 (US)

(74) Representative: Grünecker, Kinkeldey,
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Methods and arrangements for controlling access to resources based on authentication method

(57) In accordance with certain aspects of the present invention, improved methods and arrangements are provided that improve access control within a computer. The methods and arrangements specifically identify the authentication mechanism/mechanisms, and/or characteristics thereof, that were used in verifying that a user with a unique name is the actual user that the name implies, to subsequently operating security mechanisms. Thus, differentiating user requests based on this additional information provides additional control.

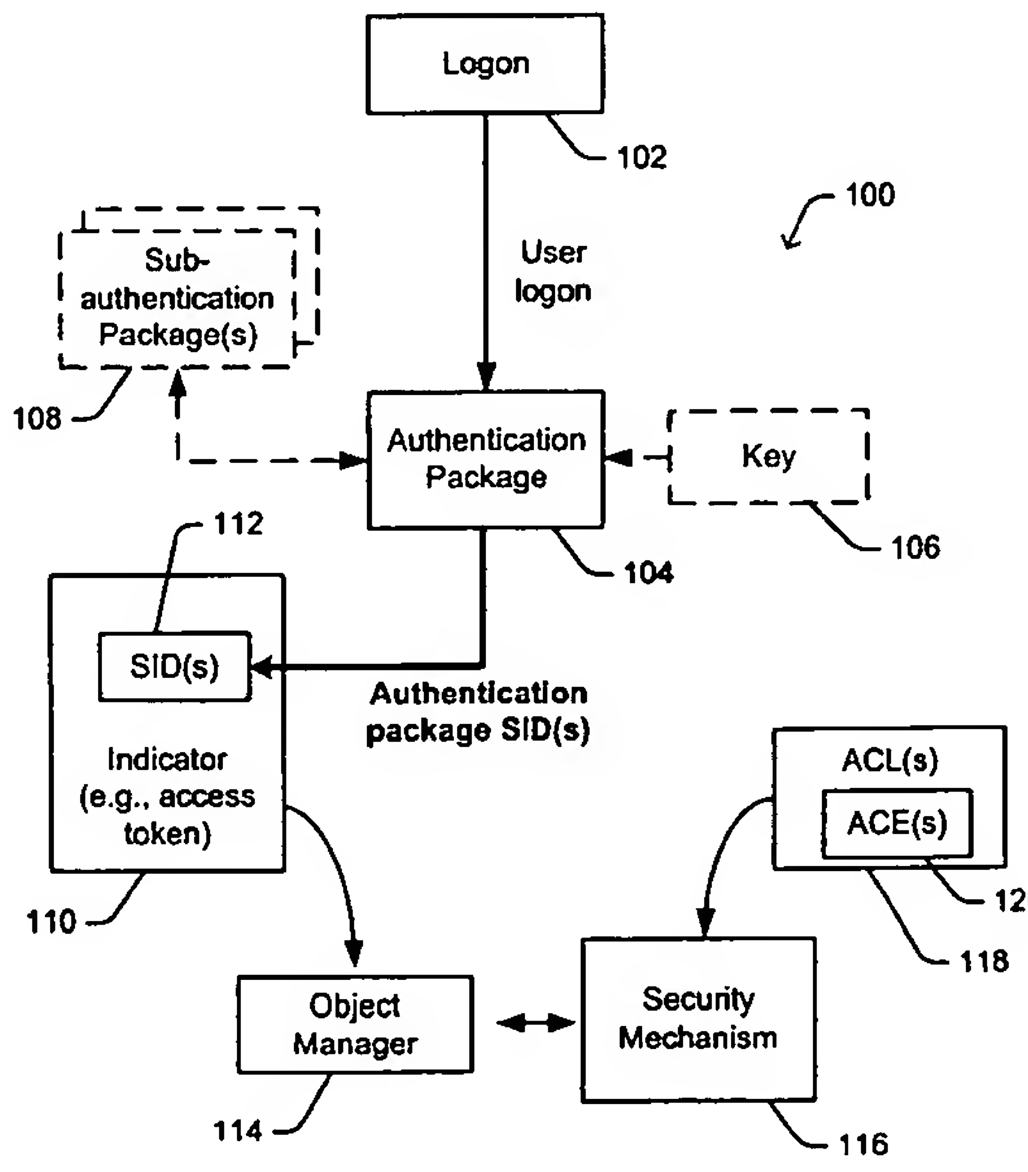


Fig. 1

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] This invention relates to computers and computer networks, and more particularly to methods and arrangements for use in controlling access to various resources therein based on authentication methods.

BACKGROUND

[0002] In the past, executable content could only be installed on a computer system by physically bringing magnetic media to the computer and having a user with the applicable privileges (e.g., administrative privileges) install it. At present, however, the Internet, intranets, wide area networks (WANs), local area networks (LANs), etc., make it very easy for ordinary computer users to download executable content, such as, e.g., ActiveX® controls, programs, and scripts. In many cases, executable content may be downloaded and executed via the Internet without the user even realizing that such an event has occurred.

[0003] Unfortunately, every so often such executable content intentionally or unintentionally destabilizes the client machine in some manner. For example, the content may prove to be error-prone and cause the client machine to crash. The content may also undermine the security of the client machine by divulging confidential information about the client/user. Although these types of computer problems have previously existed in the form of "viruses" and "trojans," the ubiquitous presence of World Wide Web (WWW) portion of the Internet has made these problems even more widespread. In general, the operating environment of most clients is not adequately protected against such unruly code.

[0004] Some operating systems already have an existing security mechanism that limits what non-privileged users may do. For example, the security system built into the Windows® NT operating system controls access to resources based on the identities of users. When a Windows NT process wishes to access a resource to perform some action, the security mechanism in Windows NT compares a client's user and group IDs and privileges associated with that process against security information assigned to that resource to grant or deny access to the resource. In this manner, unauthorized users are prevented from accessing resources and potentially causing harm, while authorized users may be limited in the actions they are allowed to perform.

[0005] There are many different authentication methods available for use in the client operating system. By way of example, a client can select among Kerberos, NTLM, Digest, Secure Socket Layer (SSL) or others that are available within the operating system. Each of these protocols is different; the differences produce varying levels of assurance as to the identity of the principals involved. Those skilled in the art will appreciate the difference between a high-assurance method such as biometric authentication, and a lower assurance scheme such as a password.

[0006] Because the eventual end-users or administrators of a computer operating system must manage access to data, protect their resources against abuse, and other tasks, these are the appropriate people to decide what assurance they require for varying tasks. Viewing a web page, as an example, may be low value enough to allow use of a low-assurance method such as a password. Updating company financial information may require a higher assurance method such as SSL. Clearly, the benefit of a consistent method, across a variety of possible applications, for controlling access would be substantial.

[0007] Hence, there is a continuing need for improved methods and arrangements for controlling access to various networked servers, devices, services, applications, etc., especially in the Internet/intranet networking arena.

SUMMARY

[0008] In accordance with certain aspects of the present invention, improved methods and arrangements are provided for controlling access to resources in a computing environment. The methods and arrangements specifically identify the authentication mechanism/mechanisms, and/or characteristics thereof, used in verifying a user, to subsequently operating security mechanisms. Thus, differentiating user requests based on this additional information provides additional control.

[0009] By way of example, the above-stated needs and others are met by a method for use in a computer capable of supporting multiple authentication mechanisms. The method includes generating an operating system representation (e.g., a security token, etc) of at least one identity indicator, for example, a user or account identity, associated with and identifying at least one authentication mechanism, and subsequently controlling access to at least one resource based on the operating system representation. In certain implementations, the method further includes generating at least one security identifier (SID) that identifies the authentication mechanism in some way, for example, by name or number and/or perhaps by measure of strength such as the type/length of an encryption process/key employed by the authentication mechanism. In other implementations, for example, the method includes comparing the operating system representation to at least one access control list having at least one access control entry therein. Here, for example, the access control entry may operatively specify whether the user authenticated by the authentication mechanism is permitted to access the resource.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] A more complete understanding of the various methods and arrangements of the present invention may be had by reference to the following detailed description when taken in conjunction with the accompanying drawings wherein:

Fig. 1 is a block diagram depicting an exemplary functional arrangement for controlling access to resources in accordance with certain implementations of the present invention.

Fig. 2 is a block diagram illustrating an exemplary computing environment, suitable for use with the arrangement in Fig. 1.

DETAILED DESCRIPTION

[0011] Authentication is basically the process of verifying that a user claiming a unique name or other identifier is in fact that user. In the physical world, this is often accomplished by using some form of documentation, issued by a trusted third party such as a government; a very common example is a passport. In the computer realm, this is often accomplished through an authentication protocol. Authorization is the determination of what a particular user or other principal is allowed to do. Authorization can take the form of limits, e.g. a credit limit on a credit card, or access controls, or, e.g. limiting what areas of a building an employee is allowed to enter. Authentication and authorization are inter-related, but are often analyzed, and even implemented, quite separately.

[0012] Two common forms of authorization within a typical computer system are name-based and identity-based. Name-based authorization essentially uses a single identifier, the user name, to manage authorization decisions. Many web sites use this form; an example is only the user has access to the user's account at a typical commercial web site. The other form, identity-based, is a richer environment. Here, the operating system maintains the identifier for the user, and possibly additional identifiers indicating groups or collections of users managed by the administrator, for use by an application. Many UNIX-derived systems expose this as a user identifier and a list of group identifiers. Windows® NT and Windows® 2000 represent this with a construct named an access token, which contains the user identifier, the list of groups, and additional restrictions and/ or privileges.

[0013] Authentication can be accomplished in either of two ways. One way is to associate a trustee name with a password on the initial connection to the data source object. The second and preferred way is to use secure access tokens or like operating system representations of some identifying indicator granted by the operating system only to authentic users/accounts. Here, the access token or like operating system representation includes one or more security identification descriptors (SIDs) that can be matched against one or more discretionary access control lists (ACLs) or the like stored in a data store.

[0014] A number of conventional authentication techniques have been implemented in authentication packages. By way of example, Windows NT and Windows 2000 provide support for the Windows NT LAN Manager (NTLM). Windows 2000 provides additional support for the Kerberos security protocol. Other well-known authentication techniques include Secure Sockets Layer (SSL), Schannel, Passport, etc. Additionally, other proprietary authentication techniques may be implemented, which are similar.

[0015] With this in mind, a generic arrangement 100 is provided in Fig. 1 that is readily adaptable to any similar arrangement. The essential functionality in arrangement 100 involves the use of an access token or like operating system representation 110 that has been further modified to include information that identifies one or more types, features and/or other aspects relating to the authentication package or packages that were used to validate the user/account. This operating system representation 110 advantageously provides for additional granularity in the overall system security model.

[0016] Thus, with reference to Fig. 1, arrangement 100 includes a logon function 102 that provides an interface with the user. The user is required to provide (i. e., input) a user/account name, password or other user/ group identifier, for example. In certain implementations, logon function 102 may further interface with logic on a smart card or like portable token device. In still other implementations, biometric information about/from the user may be gathered by logon function 102.

[0017] Logon function 102 outputs user logon information, e.g., name and password (or hash of password) to an authentication package 104. Authentication package 104 is configured to authenticate or otherwise validate that the user (based on the user logon information) is the actual user that the name implies. As mentioned, there are a number of authentication techniques in use today.

[0018] Authentication package 104 may utilize an encoding or encryption scheme that requires a key 106. In certain implementations, the trustworthiness of the authentication technique may be tied to the strength (e.g., length) of key 106. This is mentioned because this may be a security measure that is later reflected in the operating system representation 110. Authentication package 104 may also call upon one or more other sub-authentication packages 108 to verify the user logon. The use of a sub-authentication package 108 may also affect the trustworthiness of the authentication technique, and as such may be reflected in a resulting operating system representation 110.

[0019] As depicted, authentication package 104 outputs one or more authentication package SIDs 112, which are provided within an operating system representation 110. In this example, operating system representation 110 is an object that identifies the user/account as described below and is permanently attached to the user's/account's processes.

[0020] Here, operating system representation 110 includes a conventional user SID based on the logon name and/or password, etc, and at least one authentication package SID 112. Operating system representation 110 may further include other attributes such as, e.g., one or more group IDs, privileges, etc.

[0021] By providing an authentication package SID 112 within operating system representation 110, subsequent security functions will be able to further differentiate between users/accounts. This benefit and others are described below, following an overview of an exemplary security arrangement comprising an object manager 114, a security mechanism 116 and an ACL 118.

[0022] When the user's process desires access to another object it specifies the type of access it desires (e.g., obtain read/write access to a file object) and at the kernel level provides it's a corresponding operating system representation 110 to an object manager 114. The object being sought has a kernel level security descriptor associated with it that includes ACL 118. Object manager 110 causes operating system representation 110 and ACL 118 to be provided to security mechanism 116.

[0023] Within ACL 118 there is at least one access control entry (ACE) 120 that defines certain access rights (allowed or denied actions) corresponding to that entry. For example, ACE 120 may include a type (deny or allow) indicator, flags, one or more SIDs and access rights in the form of a bitmask wherein each bit corresponds to a permission (e.g., one bit for read access, one for write and so on).

[0024] As such, security mechanism 116 is able to compare the SID(s) in operating system representation 110 along with the type of action or actions requested by the user's process against the ACE(s) 120 in ACL 118. If a match is found with an allowed user or group, and the type of access desired is allowable for the user or group, a handle to the desired object is returned to the user's process, otherwise access is denied.

[0025] With the addition of authentication package SID(s) 112, security mechanism 116 may also consider the authentication package or mechanism. Thus, for example, a user that was authenticated using NTLM may be denied access to the desired object based on a deny NTLM authentication ACE 120 in ACL 118, while a another user who was authenticated with Kerberos is allowed access to the desired object. Further granularity is provided by defining different SIDs 112 and ACEs 120 based on authentication package 104, sub-authentication package 108, key 106, or any combination thereof.

[0026] Attention is now drawn to Fig. 2, which is a block diagram depicting an exemplary computing system 200 suitable with arrangement 100.

[0027] Computing system 200 is, in this example, in the form of a personal computer (PC), however, in other examples computing system may take the form of a dedicated server(s), a special-purpose device, an appliance, a handheld computing device, a mobile telephone device, a pager device, etc.

[0028] As shown, computing system 200 includes a processing unit 221, a system memory 222, and a system bus 223. System bus 223 links together various system components including system memory 222 and the processing unit 221. System bus 223 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. System memory 222 typically includes read only memory (ROM) 224 and random access memory (RAM) 225. A basic input/output system 226 (BIOS), containing the basic routine that helps to transfer information between elements within computing system 200, such as during start-up, is stored in ROM 224. Computing system 200 further includes a hard disk drive 227 for reading from and writing to a hard disk, not shown, a magnetic disk drive 228 for reading from or writing to a removable magnetic disk 229, and an optical disk drive 30 for reading from or writing to a removable optical disk 231 such as a CD ROM or other optical media. Hard disk drive 227, magnetic disk drive 228, and optical disk drive 230 are connected to system bus 223 by a hard disk drive interface 232, a magnetic disk drive interface 233, and an optical drive interface 234, respectively. These drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, computer programs and other data for computing system 200.

[0029] A number of computer programs may be stored on the hard disk, magnetic disk 229, optical disk 231, ROM 224 or RAM 225, including an operating system 235, one or more application programs 236, other programs 237, and program data 238.

[0030] A user may enter commands and information into computing system 200 through various input devices such as a keyboard 240 and pointing device 242 (such as a mouse). A camera/microphone 255 or other like media device capable of capturing or otherwise outputting real-time data 256 can also be included as an input device to computing system 200. The real-time data 256 can be input into computing system 200 via an appropriate interface 257. Interface 257 can be connected to the system bus 223, thereby allowing real-time data 256 to be stored in RAM 225, or one of the other data storage devices, or otherwise processed.

[0031] As shown, a monitor 247 or other type of display device is also connected to the system bus 223 via an interface, such as a video adapter 248. In addition to the monitor, computing system 200 may also include other peripheral output devices (not shown), such as speakers, printers, etc.

[0032] Computing system 200 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 249. Remote computer 249 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computing system 200, although only a memory storage device 250 has been illustrated in Fig. 2.

[0033] The logical connections depicted in Fig. 2 include a local area network (LAN) 251 and a wide area network (WAN) 252. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

[0034] When used in a LAN networking environment, computing system 200 is connected to the local network 251 through a network interface or adapter 253. When used in a WAN networking environment, computing system 200 typically includes a modem 254 or other means for establishing communications over the wide area network 252, such as the Internet. Modem 254, which may be internal or external, is connected to system bus 223 via the serial port interface 246.

[0035] In a networked environment, computer programs depicted relative to the computing system 200, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0036] Although some preferred embodiments of the various methods and arrangements of the present invention have been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the exemplary embodiments disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

Claims

1. A method for use in a computer capable of supporting multiple authentication mechanisms, the method comprising:

   generating at least one indicator associated with and identifying at least one authentication mechanism; and
   controlling access to at least one resource based on the indicator.

2. The method as recited in Claim 1, wherein generating the indicator further includes receiving inputs, providing the inputs to the authentication mechanism, and causing the authentication mechanism to generate at least one security identifier (SID) that identifies the authentication mechanism.

3. The method as recited in Claim 1, wherein generating the indicator further includes identifying within the indicator at least one characteristic associated with the authentication mechanism.

4. The method as recited in Claim 3, wherein the at least one characteristic associated with the authentication mechanism includes a measure of strength of the authentication mechanism.

5. The method as recited in Claim 4, wherein the measure of strength of the authentication mechanism identifies a length of an encryption key employed by the authentication mechanism.

6. The method as recited in Claim 1, wherein controlling access to the resource based on the indicator further includes comparing the indicator to at least one access control list having at least one access control entry therein.

7. The method as recited in Claim 6, wherein if the access control entry operatively specifies that the at least one authentication mechanism is permitted to access the resource, then access to the at least one resource is allowed to proceed.

8. The method as recited in Claim 6, wherein if the access control entry operatively specifies that the at least one authentication mechanism is not permitted to access the resource, then access to the at least one resource is not allowed to proceed.

9. The method as recited in Claim 6, wherein if the access control entry does not operatively specify that the at least one authentication mechanism is permitted to access the resource, then access to the at least one resource is not allowed to proceed.

10. The method as recited in Claim 1, wherein the indicator includes a security token.

11. A computer-readable medium for use in a device capable of supporting multiple authentication mechanisms, the computer-readable medium having computer-executable instructions for performing acts comprising:

    producing at least one indicator that uniquely identifies at least one authentication mechanism supported by the device; and
    causing the device to selectively control access to at least one resource operatively coupled to the device based at least in part on the indicator.

12. The computer-readable medium as recited in Claim 11, wherein producing the indicator further includes receiving inputs, providing the inputs to the authentication mechanism, and causing the authentication mechanism to generate at least one security iden-

tifier (SID) that identifies the authentication mechanism, in response thereto.

13. The computer-readable medium as recited in Claim 11, wherein producing the indicator further includes identifying within the indicator at least one characteristic of the authentication mechanism.

14. The computer-readable medium as recited in Claim 13, wherein the at least one characteristic of the authentication mechanism includes a strength characteristic of the authentication mechanism.

15. The computer-readable medium as recited in Claim 14, wherein the strength characteristic identifies a length of an encryption key employed by the authentication mechanism.

16. The computer-readable medium as recited in Claim 11, wherein causing the device to selectively control access to the at least one resource based on the indicator further includes causing the device to compare the indicator to control data .

17. The computer-readable medium as recited in Claim 16, wherein if the control data specifies that the authentication mechanism is permitted to access the resource, to which subsequent access to the resource is allowed.

18. The computer-readable medium as recited in Claim 16, wherein if the control data operatively specifies that the authentication mechanism is not permitted to access the resource, to which subsequent access to the resource is prohibited.

19. The computer-readable medium as recited in Claim 16, wherein if the control data does not operatively specify that the authentication mechanism is permitted to access the resource, to which subsequent access to the resource is prohibited.

20. The computer-readable medium as recited in Claim 10, wherein the indicator includes a security token.

21. An apparatus comprising:

at least one authentication mechanism configured to generate at least one indicator that identifies the authentication mechanism;
an access control list;
at least one access controlled resource; and
logic operatively configured to compare the indicator with the access control list and selectively control access to the resource based on the indicator .

22. The apparatus as recited in Claim 21, wherein the authentication mechanism is further configured to receive user inputs and generate at least one security identifier (SID) that identifies the authentication mechanism based on the user inputs.

23. The apparatus as recited in Claim 21, wherein the indicator further includes at least one identifying characteristic associated with the authentication mechanism.

24. The apparatus as recited in Claim 23, wherein the at least one identifying characteristic associated with the authentication mechanism indicates a measure of strength of the authentication mechanism

25. The apparatus as recited in Claim 24, wherein the measure of strength of the authentication mechanism identifies a length of an encryption key employed by the authentication mechanism.

26. The apparatus as recited in Claim 23, wherein the indicator includes a security token.

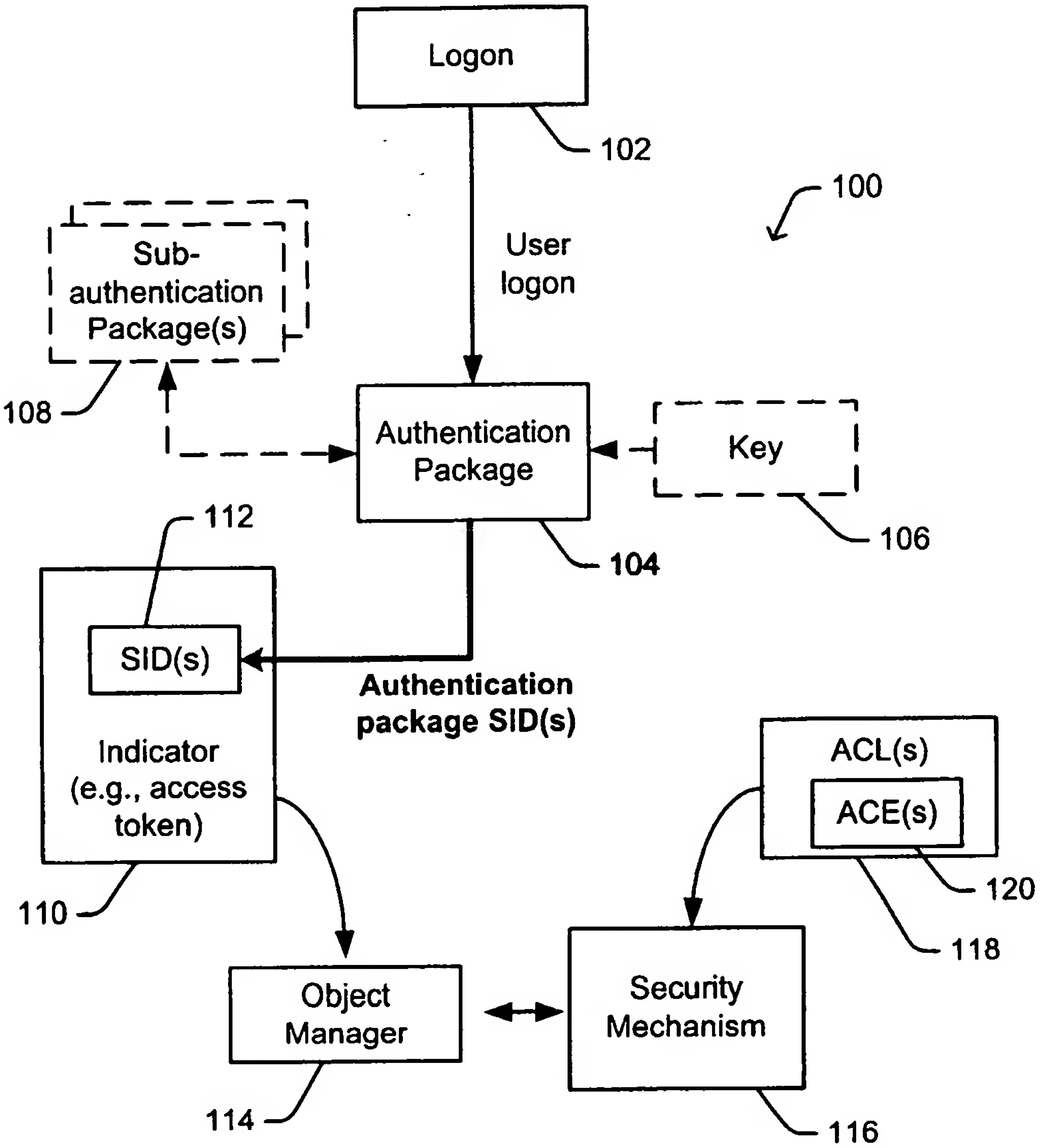

Logon
102
100
User
logon
Sub-
authentication
Package(s)
108
Authentication
Package
Key
106
112
104
SID(s)
Authentication
package SID(s)
Indicator
(e.g., access
token)
110
ACL(s)
ACE(s)
120
118
Object
Manager
Security
Mechanism
114
116

Fig. 1

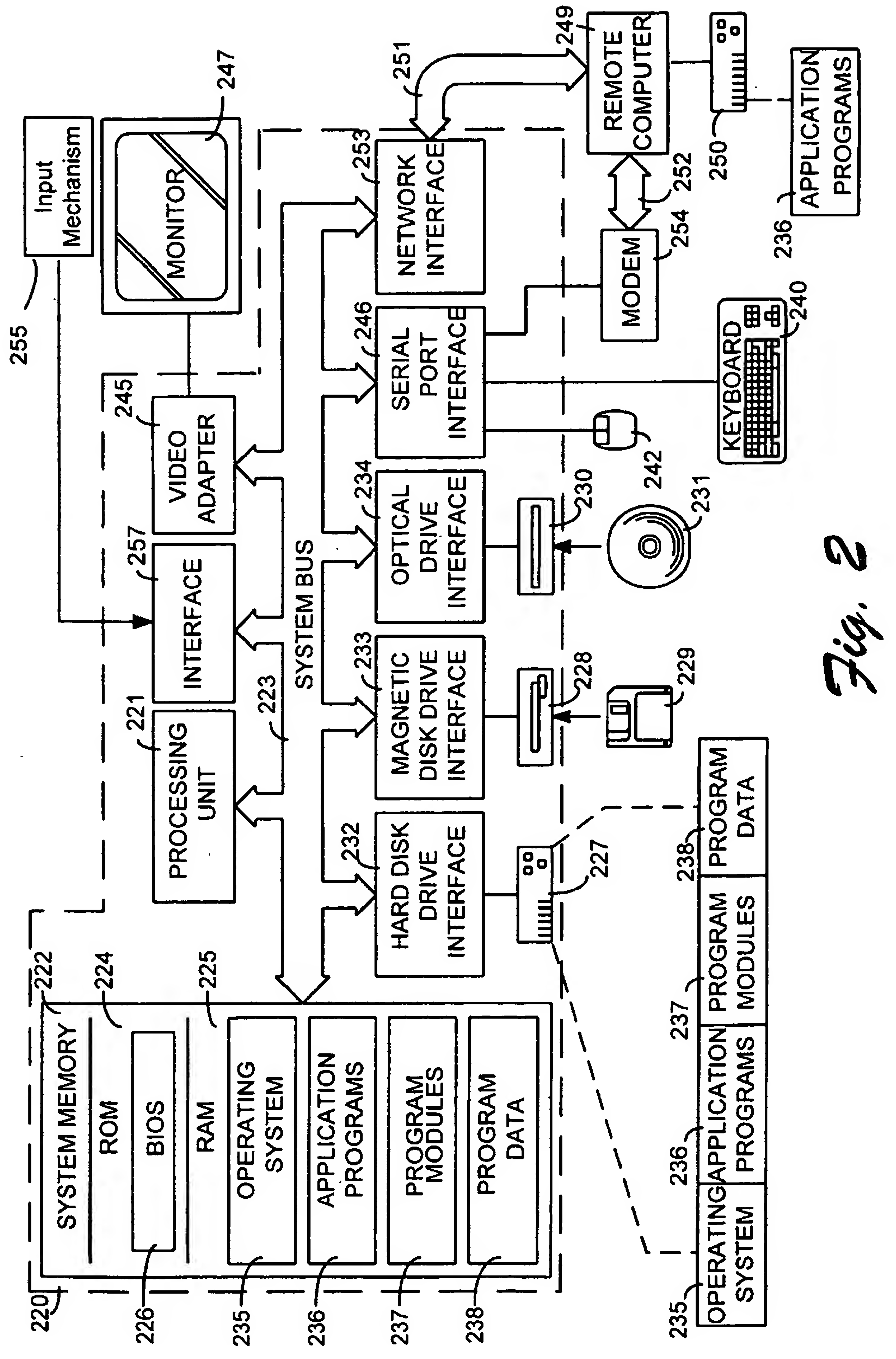

SYSTEM MEMORY
ROM
BIOS
RAM
OPERATING SYSTEM
APPLICATION PROGRAMS
PROGRAM MODULES
PROGRAM DATA
PROCESSING UNIT
INTERFACE
VIDEO ADAPTER
MONITOR
Input Mechanism
SYSTEM BUS
HARD DISK DRIVE INTERFACE
MAGNETIC DISK DRIVE INTERFACE
OPTICAL DRIVE INTERFACE
SERIAL PORT INTERFACE
NETWORK INTERFACE
MODEM
REMOTE COMPUTER
KEYBOARD
APPLICATION PROGRAMS
OPERATING SYSTEM
APPLICATION PROGRAMS
PROGRAM MODULES
PROGRAM DATA
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
240
242
245
246
247
249
250
251
252
253
254
255
257

Fig. 2

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 644 513 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94306564.9**

(51) Int. Cl.^{6} : **G07F 7/10**

(22) Date of filing : **07.09.94**

(30) Priority : **17.09.93 US 122631**

(43) Date of publication of application :
22.03.95 Bulletin 95/12

(84) Designated Contracting States :
DE FR GB NL

(71) Applicant : **AT & T Corp.**
32 Avenue of the Americas
New York, NY 10013-2412 (US)

(72) Inventor : **Mandelbaum, Richard**
15 Navajo Road
Manalapan, New Jersey 07726 (US)
Inventor : **Sherman, Stephen Andrew**
206 Chruch Street
Hackettstown, New Jersey 07840 (US)
Inventor : **Wetherington, Diane R.**
28 Woodland Road
Bernardsville, New Jersey 07924 (US)

(74) Representative : **Buckley, Christopher Simon Thirsk et al**
AT&T (UK) LTD.,
AT&T Intellectual Property Division,
5 Mornington Road
Woodford Green, Essex IG8 0TU (GB)

(54) A smart card adapted for a plurality of service providers and for remote installation of same.

(57) A smartcard that allows different Service Providers to coexist on the smartcard with none of the Service Providers, nor the owner of the smartcard, having access to the files created for, or by, each of the resident Service Providers. The operating system of the smartcard includes a root directory that is owned by the smartcard's issuer/owner, and each Service Provider is a "user" that is installed by the issuer/owner. Each such user is provided with a subdirectory of the root directory, and within the subdirectory the user creates files and subdirectories with files, as the user deems necessary. The operating system prevents all users of the smartcard, including the smartcard's issuer/owner and the smartcard's holder, from accessing any files that are owned by any other user, when that user chooses to prevent such access. This power to exclude is effected through a password file that is owned by the user and which cannot be altered by any other user, including the smartcard's issuer/owner. Optionally, the smartcard's issuer/owner is given the power to erase all files of a given user. Connection is effected with a protocol which authenticates all parties to the connection. Thus, in a connection between the smartcard and a user, the smartcard determines whether the user should be granted access, and the user determines whether the smartcard is a valid smartcard. Authentication of the possessor of the smartcard may also be effected.

Jouve, 18, rue Saint-Denis, 75001 PARIS

Background of the Invention

This invention relates to smartcards.

Advances in microelectronics have made it possible to put a vast amount of computing power in a small space. In fact, it is possible to effectively put an entire computer inside a credit card, creating thereby a "smartcard". Because of the tremendous processing and memory capabilities of the smartcard, it is expected that smartcards will supplant conventional credit cards which, typically, serve to confirm the right of the card's holder to debit a given account. Smartcards will provide a higher level of assurance that the smartcard possessor is the rightful Holder. This will solve a major problem of conventional credit cards. Moreover, smartcards will be more than an "authorizer" to debit (or credit) an account. For example, they will "carry" pre-approved credit.

To allow smartcards to fulfill their promise, Service Providers must feel secure that the computer within the smartcard cannot be employed for improper uses. A number of approaches have already been used for meeting this need. First, smartcards are provided with a power port and a single information pass-through port. Second, the computer embedded in the smartcard operates under control of an operating system which insures that instructions sent to the computer do not carry out operations that are detrimental to the card's purpose and security guidelines; i.e., only instructions that read and alter permitted data areas are allowed. Third, the issuers of today's smartcards insist on populating the card on the provider's premises and not through remote communication.

The memory in smartcards is large enough to hold the programs and data of a number of service providers. That is, there is sufficient memory to allow, for example, VISA, AMERICAN EXPRESS, and MASTERCARD to coexist on a single smartcard. Alas, smartcards have yet to be developed that, in a commercial sense, have succeeded in carrying the services of more than one Service Provider. It is believed that the reasons for this state of affairs is a number of security problems have not been solved. One problem, for example, arises in connection with who is the card's owner, and what powers does the owner have over all the files in the smartcard's memory. Stated in commercial terms, the question is to what extent does the owner of a smartcard (who may also be a Service Provider) have powers over the smartcard that are inconsistent with the security that other Service Providers seek. This is a *trust* issue.

A second issue relates to remote provisioning. Particularly, it is undesirable to require the smartcard holder to have a service installed only by brining the card to the provider. It is also undesirable to require surrender of the smartcard when one of the services on the smartcard is to be canceled. Rather, it is desirable and perhaps even essential for commercial success, to allow remote provisioning.

When the remote provisioning issue is solved, a third issue relates to the need to reuse space in the holder's smartcard as old services are canceled and new services are installed.

A fourth issue relates to the commercial conflict between competitive services, and the desire by some providers to restrict access by their customers to competing services.

Summary of the Invention

The issues described above are resolved with an operating system that allows different Service Providers to coexist on a smartcard with none of the Service Providers, nor the owner of the smartcard, having access to the files created for, or by, each of the resident Service Providers unless authorized in advance.

The operating system of the smartcard, somewhat akin to the UNIX® (registered trademark of UNIX System Laboratories) operating system, includes a root directory that is owned by the smartcard's issuer/owner, and each Service Provider is a "user" that is installed by the issuer/owner. Each such user is provided with a subdirectory of the root directory and within the subdirectory the user creates files and subdirectories with files, as the user deems necessary.

The operating system prevents all users of the smartcard, including the smartcard's issuer/owner and the smartcard's holder, from accessing any files that are owned by any other user, when that user chooses to prevent such access. This power to exclude is effected through a password file that is owned by the user and which cannot be altered by any other user, including the smartcard's issuer/owner. Optionally, the smartcard's issuer/owner is given the power to erase all files of a given user.

The operating system also includes means for digital signature-supplemented communication as well as for completely encrypted communication. This capability imparts confidence in remote communications, which permits remote provisioning, effective maintenance of a database that keeps track of all services contained in each smartcards, and re-provisioning of a smartcard in case of loss or general failure of the smartcard.

Brief Description of the Drawing

FIG. 1 depicts a tree structure of the UNIX operating system;

FIG. 2 presents the tree structure of a smartcard operating system;

FIG. 3 shows a log-in protocol between a smartcard and its issuer/owner;

FIG. 4 illustrates a protocol involving a smart-

card, the issuer /owner and a Service Provider;
FIG. 5 presents a protocol for a smartcard obtaining services from a Service Provider;
FIG. 6 presents a protocol involving a smartcard, a Visitor user and a Service Provider; and
FIG. 7 presents a protocol between a smartcard and a Visitor user, without connection to a Service Provider;
FIG. 8 depicts an arrangement for remote provisioning of smartcards using the telecommunication network; and
FIG. 9 presents a flowchart of an operating system command for drawing upon a value stored in a service provider's file.

Detailed Description

A number of smartcard operating systems are already known. One example is U.S. Patent 4,816,653, issued to Anderl et al on March 28, 1989. The operating system described below has many similarities to that operating system and to the well known UNIX operating system. A brief description of some well known aspects of the UNIX operating system will help in understanding the smartcard operating system disclosed herein.

The UNIX Operating System

The UNIX operating system comprises a collection of files. Some are files that primarily contain information about related files, and they are called directory files or directories. Others are files that contain user data, and they are "normal" files. Also in the UNIX operating system, a user can be the "owner" of the file, can belong to a specified "group" recognized by the file, or can belong to "other". Each file contains a data portion that specifies the file characteristics, such as ownership, information access capabilities relative to the three types of users, etc. The owner of the file can change all file characteristics.

Architecturally, the first and primary file is a root directory file. The user who is the owner of this directory is, in effect, the owner of the entire operating system. This user can create other files which are pointed-to by the root. file. These files, which can be other "directory" files as well as "normal" files, are considered to be "below" the root directory, in a tree-like structure.

In many UNIX operating systems, one of the directories below the root is named "etc.", and it has a file below it that is designated "passwd". The full address, or path name, of that file is "/etc./passwd" (the file "/" at the beginning of the path name designates the root address). The "etc." and the "passwd" files are owned by the system administrator, typically called "Root", who is the also the owner of the root directory. The "passwd" file contains an encrypted representation of Root's password, and Root's access to the operating system is allowed only after Root logs in by providing the password. The presented password is encrypted and compared to the encrypted password stored in the "passwd" file. When the comparison is favorable, the user is accepted and granted permission to access other files; i.e., the user is "logged in".

Multi-user capability is provided by allowing Root to create a subdirectory below the root directory and to assign ownership of that subdirectory to another user. Root can then install a password for that user in the "passwd" file and allow the user to enter the system at that subdirectory file when that user presents his/her password. The user has the ability to modify his/her own password, but only through a command provided by the operating system. That password resides in the system only in encrypted form and only in the "passwd" file. This architecture is depicted in FIG. 1.

The log-in process can be summarized as follows. A computer operating under the UNIX operating system begins by executing a loop that scans the computer's input port. When connection by a user is detected, control is transferred from the loop to a program that begins interactions with the user. The program sends a "login:" message to the user and waits for the user's response. The user identifies himself/herself, for example, by returning the string "htb" and that identifies the user to the operating system. The program then continues with the challenge message "Password:" and the user must supply a password string. The program encrypts the password string and compares it to the encrypted password that is found in the "/etc./passwd" file for the identified user. When the match is positive it is determined that the user is *bona fide,* and control passes to a file owned by Root (typically named ".profile"). That file sets various parameters for the user and passes control to another file that is owned by the user (typically also named ".profile", but this file is located in the directory owned by that user). Whatever instructions are found in the user's ".profile" file are executed and the computer is placed in a loop, awaiting further instructions from the user.

Root is the owner of all files that comprise the operating system, as well as of the "passwd" file. Therefore, Root can modify any and all files and is, therefore, a "super user". It is important to note that even files that are not owned by Root are nevertheless subject to Root's commands. Reason: Root has the power to change the "passwd"file as well as the files the control Root's capabilities generally That capability gives Root the power to change the password itself and, therefore, Root can always make itself the owner of a file. Hence, it makes sense to let Root have all owner powers in the first instance. In short, Root has absolute control and total knowledge over all files in the system.

Aside from being able to log in (by providing the correct password), users are granted the ability to read files, write into files, and execute files - i.e., pass program control to files. (Without the power to pass program control to a specified file nothing can be done, for executing a program is noting more than passing control to a file.) Since Root has access to all files in the system, it follows that Root can read, write, and execute all files.

All system-provided instructions in the UNIX operating system are merely files that can be executed, and those files can be located in any directory -- as long as the system knows where those files are found. As stated earlier, Root owns all those directories and those files. Since Root controls the read and execute permissions of all those directories and files, it follows that Root can restrict anyone (including itself, if that were desired) from executing any file, simply by limiting the permissions of that file. That gives Root the power to create customized sets of files whose execution is blocked to particular groups of users. In other words, Root can create various restricted operating systems, or "restricted shells", that encompass less than all of the commands available on the system.

The Smartcard Operating System

The absolute power that Root has in the UNIX operating system makes it unsuitable for smartcards. While it is patently clear that providers such as VISA, MASTERCARD, and AMERICAN EXPRESS will not allow each other to be the Root, it is also quite likely that, absent demonstrably sufficient security means, they would not want anyone else to be the Root either. That is part of the problem that has been blocking the smartcard from enjoying the commercial success it deserves.

FIG. 2 illustrates a structure that responds to this sensitivity of Service Providers. According to the structure of FIG. 2, Root owns the root directory and any number of files (directory files or normal files) that it wishes to create. For example, FIG. 2 includes a root directory file 10 and below it there are ".profile" file 11, "passwd" file 12, "log" file 17, "filex" file 13, "filey" file 14, and "ID" file 18. A number of subdirectories are also found below root, with each being used as the "HOME" directory of a user (Service Provider). For example, FIG. 2 includes a directory file 15, named "htb" (the smartcard's Holder), a directory file 20 named "bankA", and a directory file 25 named "airlineA". Each one of the directories includes a "passwd" file (16, 21, and 26, respectively) below the associated user's HOME directory, as well as a "profile" file. This placing of the password files has some advantages by it is not a requirement. Importantly, ownership of each such password files is assigned to the user associated with that file and the directory above it. It may also be advantageous to grant ownership of directories 15, 20 and 25 to the respective users.

FIG. 2 includes one additional important directory (and a user). That is the "Visitor" directory 30, which is the entry point for non-Service Providers who wish to interact with the smartcard.

The FIG. 2 file architecture is coupled to an operating system that differs from that of the UNIX operating system primarily in that the operating system of the FIG. 2 structure does not allow Root the ability to modify files that it does not own. To insure that this capability is not circumvented by Root, the operating system does not allow Root to modify some of the files that *define* the operating system (in a sense, Root does not own those files). One means for achieving the latter result is to encase those non-Root-owned operating system files in a read-only memory (ROM). At the very least, the ROM contains the commands/modules/files that effect writing to a file. More specifically, the writing to a file is restricted to whatever the owner of the file specifies (the owner of a file is, initially, the user that creates the file), and Root is treated as merely another user. Commands that effect writing to a file are operating system commands that, for example, move files, copy files, save files, change file attributes (e.g., ownership), and rename files. Other items that may be installed in the ROM, or more typically in a "write once" memory (because they are unique to each smartcard), are the Root password and the smartcard's ID information (i.e., files 12 and 18) The ID information may be simply an arbitrary string, or it may include the Holder's name. Including the Holder's name is probably preferred by merchants who will get the ID information. Actually, both the Root password and the smartcard's ID can be encased in the file that establishes the Root directory (i.e., in block 10). In FIG. 2 these are independent files, however, to more clearly illustrate the concepts.

One file-writing power that is granted to Root in some embodiments is the power to delete any file in its entirely (and in the process, effectively deleting any file that the deleted file points to). This includes directory files and normal files and it applies to files that Root owns and to files that Root does not own. Such a capability may be given in embodiments where memory space is to be reused when a given Service Provider is no longer providing services to the smartcard's Holder.

Another difference between the operating system of FIG. 2 and that of a standard UNIX operating system is that the former includes an encryption key pair that is installed in a file owned by Root (e.g., in "filex" 13), and that key pair is unique to each smartcard. The pair includes a private key, f, that is kept secret by the smartcard, and a public key, g, that the smartcard does not care to keep secret. Of course, both keys are initially known to the smartcard's owner/issuer, who is also the Root user (i.e., super user) of the smartcard, but Root need not keep the private key (and probably would choose to destroy that knowledge). This pair of keys can also be "burned" into an appropriate memory, such as the memory containing Root's password, or included in the file that defines the root directory. More about public key encryption is found below.

The fact that the password of a user's directory is stored in a file that is owned by the user is a key difference between the UNIX operating system and the operating system shown in FIG. 2. The fact that these passwords can't be read by anyone other than the files' owners permits storing the passwords in an unencrypted form. Combined with the restriction on writing, this organization prevents Root from becoming the owner of any file (normal file or directory file), and thus prevents Root from circumventing the permissions set by the file's owner. This key difference allows one user's files to be completely opaque to Root as well as to other users. Thus, the FIG. 2 arrangement overcomes the "trust issue" between the providers of services and the smartcard's issuer/owner.

Transactional Security

The next issue that must be addressed is transactional security of the smartcard. This concept encompasses the measures employed by the smartcard's operating system and by the agreed upon communication protocols to ensure that unauthorized transactions do not occur which would adversely affect the HOlder or any of the Service Providers. This includes activities by Root, the Holder, any of the Service Providers, any Visitor user, or an interloper. (An interloper is a party that interjects itself into a communication session between a smartcard and another party and substitutes its messages for the true messages.)

One way to thwart interlopers is to construct messages that include a date and time stamp, with at least that portion of the message being encrypted. Alternatively, the entire message can be encrypted. Also, wherever necessary, the communication protocol can require a confirmation sequence (which differs from session to session) to be exchanged between the parties. It is also a good general approach to minimize the flow of sensitive information in the clear (i.e., without encryption). These techniques are employed in the log-in and communication protocols described below.

Encryption

The field of encryption is not new. What follows is merely a summary of two encryption techniques that may be employed in connection with the smartcard disclosed herein.

As is well known, the "shared secret" approach to encryption calls for the two communicating parties to share a secret function, f. The party wishing to transmit a message, m, encrypts the message with the secret function to form an encrypted message f(m). The encrypted message is transmitted and the receiving party decrypts the received signal by forming the function f(f(m)). The function f is such that discovering the message m from f(m) is computationally very difficult, but applying the function twice recovers the original message; i.e., f(f(m))=m.

The "shared secret" approach to encryption is very good, but its weak link lies in the need to communicate, i.e., or share, the secret function. If an eavesdropper obtains the shared secret during that singular communication session when the function is transmitted, then it is no longer secret.

In public key encryption, each party maintains one member of a pair of keys, f and g. More particularly, one party keeps one key (f) secret, but makes the other key (g) known to all. Thus, the key g is "public" and the key f is "private". The pair, f and g, is such that

1. g(f(m))=m,
2. even when g is known the function f cannot be determined, and
3. it is computationally infeasible to determine the message m from f(m).

Whereas the public key approach solves the key distribution/administration problem described above, it does have a disadvantage in that public key encryption and decryption is slower (requires more computation time) than the "shared secret" approach.

As it relates to smartcards, speed of communication has different levels of importance, based on the kind of party that is communicating with the smartcard. With respect to the smartcard's issuer/owner and the service Providers, low speed is not a major disadvantage because it is expected that such communication will be rare and, therefore, processing time is not "of the essence". In communication with others, however, (i.e., merchants that log in as the Visitor user), speed is important.

The speed issue is resolved, where necessary, by combining the "shared secret" approach with the public key approach. That is, when communication is initiated, the public key approach is used to communicate a temporary "shared secret" between the smartcard and the merchant. Specifically, the party having the public key suggests a "shared secret" and communicates it to the party having the private key. Thereafter, the faster, "shared secret", approach is used to encrypt the entire messages.

Alternatively, a certification approach may be used (using the shared secret). In a certification approach, the message is sent in the clear, and is appended, or "signed", with a "digital signature". A "digital signature" is a hashing of the message (e.g., adding the ASCII codes of the characters in the message, in a selected modulus) that is encoded. Of course, in applications where it is assured that an interloper cannot substitute the true data with false data the information can be sent in the clear (probably, following a verification process using the public key).

Use of the public key approach solves most of the key-administrations concerns. It still leaves the question of the initial *knowledge* of the public key by the party wishing to communicate with the smartcard, but that is a non-problem since the smartcard itself can provide that information.

Log-in by Root and Installation of a Service Provider/User

Because encryption ensures secure communication, the smartcard's issuer/owner can have confidence in remote installation of services. Of course, the issuer/owner (i.e., Root) must first log in into the smartcard. A protocol for the log-in is presented in FIG. 3, and a protocol for service installation process is presented in FIG. 4. The physical remote, connection that is possible with the smartcard disclosed herein is shown in FIG. 8.

As depicted in FIG. 3, the process begins with the smartcard's possessor (P) being authenticated as the *bona fide* Holder (H) of the smartcard (S). As shown in FIG. 3, the process begins with a prompt from the smartcard, and anentry of the possessor's PIN (Personal Identifical Number) into the smartcard. The use of the smartcard's processing power to authenticate the possessor has an advantage in that it requires no communication of the PIN string to any equipment that might capture the PIN. Even in applications where P and S are at a merchant's premises, it is possible for the merchant to possess a stand-alone piece of equipment that interfaces with the smartcard in a secure manner. This equipment may be battery operated, with a keyboard and a display and certified to include no other ports, no processor, and no writable memory. In operation, P would insert S into this stand-alone equipment, input the PIN via the keyboard, and the smartcard would determine whether the PIN is correct. It would then output an "OK" message through the display, if appropriate.

When such stand-alone equipment is unavailable (or when the communication is remote as, for example, when a "dumb" card reader is used at the possessor's home), the submitted PIN is processed in the smartcard and the "OK" message from the smartcard (to the merchant's equipment) is "time stamped" and encrypted. This suggests that the P's confirmation as H must be postponed until after the appropriate encryption keys are established and date and time information is imparted to S (this is not the approach shown in FIG. 3).

Returning to FIG. 3 generally, after the *bona fide* of H is established, S verifies that the user logging is a valid user and the user confirms that S is a valid smartcard. More specifically, the protocol of FIG. 3, in its entirety, proceeds as follows:

a. S prompts for an input and P provides a PIN string. (Within the smartcard the PIN resides in a Root-owned file that is open for the Holder to modify, for example, file 14 in FIG. 2). S compares the provided PIN string to the stored PIN string, and, if the comparison is positive, then P is confirmed as H.

b. Once H is confirmed, attention can turn to the communication between S and O. S identified itself by providing to O its ID number and a password challenge in the form of a random string, RND1.

c. O encrypts RND1 with O's password to form string K_1(RND1)and returns it to S. This form of password response obviously changes from session to session and ensures that the true password of O is not snared by an interloper. There does remain the question of where O keeps the passwords of all the smartcards that it owns, and how secure such a database is. However, there is actually no need for O to keep a database of passwords. All that O needs is a single <u>seed string</u> which, when processed with the data supplied by S, yields the smartcard's password. That data is the ID information.

d. Since the string submitted by the smartcard will always be either the same or unknown to O beforehand, an additional authentication step may be desired to ensure that the initial string (ID, RND1) is not a replay of a recording. This is achieved by O sending a challenge message, RND2, to S comprising, for example, its ID and a random string.

e. Based on the ID contained in the RND2 string, S determines that O is the user, obtains the necessary keey (e.g., O's password), and decrypts K_1(RND1). When the decryption results in RND1, S determines the O is *bona fide*.

f. Thereafter, S encrypts string RND2 with S's Root password and forwards the resultant string, K_1(RND2), to O.

g. O decrypts the K_1(RND2) response, and if the resulting string is RND2 then O is satisfied that S is valid. This ends the log-in process, with O presenting a prompt to S and standing ready to accept requests for service.

It may be noted that the "log-in" process described above appears to be different from the familiar log-in process where the computer into which access is directed controls. The computer asks for an initial identification of the user, and then asks for a password. Based on that initial identification, the computer knows what password to expect. Here, the smartcard appears to be in control, in the sense that it initiates the communication (with O); but instead of asking for an initial identification -- to get information --, it provides information -- the ID and RND1. That raises the question of whether the response from O is the initial identification, or the password; and if it is the password, then how does S know whether the password is correct. The answer is that the response from O serves three purposes: it identifies itself in the sense of the initial identification (by the ID contained in RND1), it authenticates itself by using the correct key to encrypt RND1, and it challenges S by RND2 to be returned in an encrypted mode.

Once O is logged in, H can communicate a request for installation of a service offered by a Service Provider (SP). The communication regarding the particular service requested to be installed by O may involve interaction with a human, but it can also be automated. For example, H can communicate to S the service that is desired, and S communicates with O. FIG. 4 presents a protocol for installation of service.

a. H forwards a service request to S

b. S encrypts the request and forwards it to O. The electronic communication between O and S can be encrypted with the private key member of the public key within S, with S sending its public key to O. Alternatively, the communication can be encrypted with a "shared secret" of the smartcard. The Root password may be selected for the latter, or a temporary "shared secret" can be offered by O to S (using public key encryption, as described above). In FIG. 4, the Root password is used for encryption, creating the request string K_1(REQ).

b. Knowing the desired service, O contacts SP to verify that SP consents to offer service to H

c. If provision of service is agreeable to SP, O selects a temporary password, informs SP of that password (probably through encrypted communication), and proceeds to create in S a directory and a password file for SP.

d. When the password file is set up for the SP user, the temporary password is sent to S (communicated in encrypted manner, as described above) and ownership of the directory and the password file is transferred to SP (this password can serve as the "shared secret" key in future communication sessions with SP). Also, the rest of the application software that SP requires can be installed at this time, with O transmitting those files in encrypted mode. Alternatively, it may be arranged that no application software is installed by O.

e. At this point H is informed to contact SP for a final set-up.

f. H sets up a communication path between S and SP, employing a log-in sequence as shown in FIG. 3 but using the temporary SP password as the encryption key.

g. Once the log-in to SP is established, S sends out a request for service, and SP responds by installing a new password, as well as data and whatever files are necessary which were not installed by O, and data. This completes the service installation process.

Provision of Service by a Service Provider

As indicated above, a Service Provider is simply a user having an assigned directory in the smartcard. The Service Provider logs in when the Possessor establishes communication between the smartcard and the Service Provider. As before, there are three elements to the log in protocol:

(1) SP wishes to establish that P is H,

(2) S wishes to determine that the logged in user is the true SP, and

(3) SP wishes to determine that it is communicating with a valid S.

These three elements are carried out in the protocol disclosed in connection with FIG. 3. Only after a successful log-in, can a service request be advanced. A service request may comprise, for example, H requesting SP (e.g., a bank) to install "money" into S, filling the "electronic purse" of S. The filling of the electronic purse may simply be installing a value in a file owned by SP. This is shown in the flowchart of FIG. 5.

Interaction with Merchants

It is expected that, by a wide margin, the smartcard holder will mostly wish to have the smartcard interact with merchants who are Visitor (V) users. The arrangement disclosed above permits such interaction in two ways: direct interaction between the smartcard and a merchant, and a three way interaction involving the smartcard, merchant and the Service Provider. The protocol for the latter approach, shown in FIG. 6, may be as follows:

a. P sets up communication between S and V (by handing S to V or by remotely connecting S to V).

b. S prompts for an input and P provides the PIN string. When that checks out, S determines the P is H and proceeds with the standard "log-in" sequence, sending its ID information and RND1.

c. V sets up a communication path with SP, identifies itself to SP, and relays the ID information and RND1.

d. Given the ID information, SP determines its password (perhaps also using a seed string combined with processing) and encrypts RND1 with that password. The resulting string, K_2(RND1), is sent to S, together with a random string RND2.

e. S determines whether SP used the correct password in forming K_2(RND1) and, if the conclusion is positive, encrypts RND2 and forward the result, K_2(RND2), to SP.

f. When SP confirms that S used the correct password to encrypt RND2, it sends a prompt to V to inform the merchant that the it can proceed to request usage of S.

g. V requests action from SP (such as deleting a value from H's account with SP, or such as modifying a value in a file residing in S and owned by SP.

h. SP fills that request and, if necessary, sends an appropriate command to S, encrypted with the SP password.

When it is desired to have the smartcard interact directly with the merchant (or a merchant in concert with the merchant's bank, or some other entity that provides a service to the merchant and stands "in the shoes" of the merchant) a mechanism needs to be established for allowing parties who do not have a pre-established relationship with the smartcard to log in into the smartcard. The "Visitor" user directory serves that need, and that user has no password. That means that the Visitor user is a very insecure user, so its access must be strictly controlled.

One question that needs to be answered, for example, is whether such a Visitor user will have access to application files (programs) of only the Service Provider specified by the merchant, or to application files of all Service Providers.

If access is to be granted to application files of all Service Providers, then the simplest arrangement is for Root to establish a Visitor user directory with no password and with a restricted shell which allows the Visitor user to execute only a limited set of operating system commands; i.e., with the variable PATH is set up to contain one directory owned by Root (which includes only few operating system commands) and the SP directories (or chosen subdirectories of the Service Providers/users) which contain the executables to which the SPs wish to grant execution access to Visitor users.

If access is to be granted to application files of only a specified SP then, of course, that SP must be specified and means must be provided to include only the executables of the specified SP. Again, that is easily accomplished with a restricted shell, where the PATH variable includes the directory (or chosen subdirectory) of the specified SP. The protocol, depicted in FIG. 7, may be as follows:

a. S prompts for an input and P provides the PIN string. When that checks out, S determines the P is H and proceeds with the standard "log-in" sequence, sending its ID information and RND1.

b. Since V does not have any passwords, it merely returns the string RND1.

c. By this response S recognizes that the user is a Visitor user and sends out its public key, K_{pu}. (The public key could have been sent as part of the ID information.) At this point S can also send a "digital signature" that is derived from a message that contains the public key, the ID information and RND1. S can also send an encrypted string that constitutes a proposed "shared secret" (not shown in FIG. 7). The digital signal is encrypted with the public key.

d. V deciphers the "digital signature", using the provided public key. If the deciphered "digital signature" matches the appropriate string then V sends out RND2.

e. S encrypts RND2 with its private key and responds with K_{pr}(RND2).

f. V decrypts this message with K_{pu} and if V obtains RND2 then V is satisfied that it is communicating with S.

g. V sends time and date information to S, encrypted with K_{pu}, and S returns a prompt.

h. V transmits a request to S (identifying the action V seeks and the SP that is to be employed), also encrypted with K_{pu}, and S responds with an authorization to contact the specified SP. The authorization is encrypted with the private key, K_{pr}.

Typically, the merchant wants to receive funds that belong to H, in exchange for goods or services provided by the merchant. As described above, it is quite possible for a Service Provider, such as a bank, to install an "electronic purse" that will hold a value. This value is in a file owned by the Service Provider.

The merchant wants access to this file, and SP (in concert with the H) is willing to grant access to this file, but only in a very limited and strictly controlled way. Thus, SP creates a file with a prescribed name that is expected by the operating system, populates this file with a value, and a specific operation system command (that is not owned by Root) accesses the file and deducts a sum from the value found in the file.

A flowchart of such a command is illustrated in FIG. 9. the command starts at block 200 by perusing through a file (of a prescribed name) in the Visitor user directory. The file must contain four entries, separated, for example, by a newline character, and the operating system assumes that the four entries comprise a) date and time, b) merchant's ID, such as name address, and perhaps a code, c) the sum of money that is to be deducted, the d) the Service Provider whose "electronic purse" is to be used.

When that file does not exist or doesn't have the rquired number of entries, control passes to block 210 which informs the merchant (Visitor user) of the deficiency. When the file does exist, the command reads the value in the "electronic purse" file of the Service Provider (SP) in block 220. This file is a file that has a prescribed name. Block 230 evaluates whether the sum that the merchant wishes to withdraw is greater than the value in the electronic purse. If it is, control passes to block 240 which constructs a rejection message and forwards it to the merchant and to the log file within the smartcard. When sum is lower than the value, control passes to block 250 which checks the log file for various indicia of fraud. This may be a separate command that is called by the command being executed. As depicted in FIG. 3, block 250 can result in three types of outputs: one that suggests a potential fraud condition (e.g., this merchant has used the smartcard more than a prescribed number of times within a preselected time interval, the data and time given by the merchant is earlier than the latest time found in the log file, etc.); another that responds to a threshold file provided by the SP which directs the merchant to conference the SP to the transaction; and the third that indicates a normal condition.

The potential fraud condition is handled by information being stored in the Holder's log file (block 260) and control then passes to block 240. The ifnormation stored identifies the merchant, what was attempted to be withdrawn, the reason for the rejection, etc. This provides the Holder with the information necessary to interact with the issuer/owner of the card and with government authorities, as necessary. If desired, the smartcard is disabled when a fraud condition is suspected.

When a threshold set by SP is exceeded (e.g., SP desires withdrawal authorization in excess of $1,000 to be granted "in real time"), a message is constructed in block 270 and control passes to block 280.

Block 280 is also arrived at directly from block 250 when a normal condition is indicated. Block 280 increments the sequence number found in the smartcard's log file and deducts the sume desired by the merchant from the amount in the value file. Thereafter, block 290 creates a string that comprises the new sequence number the data and time, the merchant's identification information, the sum, and the SP. Block 300 creates a digital signature of the string and block 310 creates a message that comprises the message constructed in block 220, the string constructed in block 300, and the digital signature. Finally, that message is sent to the merchant and to the smartcard's log file.

The merchant's equipment will do one of two things. When a message to conference SP is found to be present, the merchant's equipment connects itself to SP and forwards the message created in block 310. The merchant can then get immediate credit for the sum (provided, of course, that based on the signature the message is concluded to be valid). When the message received by the merchant does not include a message constructed by block 220, then the merchant can simply store the authorization string, collect such authorization strings over a chosen time interval (for example, the entire work day), and then forward the authorization strings to the appropriate SPs.

The authorization string is shown encrypted with the private key of S, but it can also be encrypted with the password of the specified SP. The authorization string must be robust enough ensure that the merchant does not merely duplicate it a number of times and send it to the SP. This can be accomplished in a number of ways, including having the date and time stamp, an indication of the "before" and "after" values in the value file, having a sequence number supplied by S, etc. Since this authorization string is not decipherable by V and hence unalterable, security is maintained.

Smartcard Issuer/Owner as a Service Center

One aspect of the arrangement disclosed herein is that the smartcard's issuer/owner (O) has a general knowledge of, and control over, the service providers whose "applications" are present on the smartcard. First, O controls the establishment of a Service Provider's directory. Second, O can delete any directory at the holder's request, or whenever O gains access to the smartcard (with, or without, the holder's consent). Third, O is the only party who knows the identity of all the Service Providers who share the smartcard, and various particulars about those Service Providers. Fourth, through the operating system's design, O can control the amount of memory that each Service Provider has access to, and thus can control the number of Service Providers that can "coexist" on a smartcard. Fifth, O can define a Service Provider's grouping for particular types of transaction. Sixth, for the privilege of being on the smartcard O can charge each such Service Provider in proportion to the space occupied by the Service Provider.

As can be appreciated from all of the above, a number of benefits accrue from this arrangement. One that hasn't been mentioned before is that O has the power to "fix" a defective card and *reinstall* all of the services on it (typical power of an owner). Conversely, O has the power to delete all directories. This latter power is exercised when it is determined that a security breach has occurred.

With regard to security, there are four forms of attack that need to be considered: one is when an interloper tries to become Root, another when the interloper tries to become a Service Provider, a third when a party (Root, a Service Provider, and interloper, a Visitor, or the Holder) tries to do more than is permitted, and a fourth when the possessor is not the *bona fide* Holder.

With respect to the first form of attack, it is the Root password which is the first and primary sentry. It is an effective sentry in that the operating system is set up to completely disable the smartcard when a log-in as Root is attempted but fails. For example, all directories can be erased.

An attempt to log in as a Service Provider should be only slightly more forgiving. Thus, it can be arranged for a counter to keep track of failed attempts to log in as a Service Provider. When the number of failed attempt exceeds a preselected value (for example, 4) the smartcard again disables itself. In such situations it may be desired to direct the smartcard disablement only to the directory of the Service Provider who was the object of the attack, or to all of the Service Provider directories but not to the Root directory.

As stated above, the most numerous contacts with the smartcard will be by Visitor users. While these contacts need to be flexible, they also need to be vigilant. Whereas in the UNIX operating system an attempt to execute a command that is not found in the PATH results in a benign message, the smartcard needs to monitor these attempts to access impermissible commands. Again, a counter can be employed and when a preselected count is exceeded, communication with the Visitor can be terminated, a message stored in the smartcard, and the card disabled to everyone except the Holder. The message, which would be stored in the holder's directory, would comprise the particulars of the aborted transaction. Similar actions would be taken if the Holder attempt to execute impermissible commands, but the diagnostic message would then be written to a Root-owned file.

Another security measure might even involve valid transactions by the Visitor. As described above, one of the files owned by Root is a log file, which maintains a record of all transactions carried out by the smartcard. This file can be checked to disallow a particular Visitor user, or all Visitor users, when particular circumstances appear to be present, such as too many transactions by one visitor in a given time interval, too many transactions in a given time interval, etc.

A slightly different security problem manifests itself when the parties making contact with the smartcard are OK, but it is the possessor of the card who is the problem. Here, it can easily be assumed that the parties interacting the smartcard would like to cooperate in the prevention of the smartcard's use; at that point, and henceforth. This can be accomplished in a number of ways. For example, when, during the login sequence it is determined that the ID provided by the possessor is wrong (because, for example, the smartcard is a stolen one), the merchant can execute a command that writes a message to a file belonging to Root and disables the card. In such an event, the only way to recover the card is to contact Root. When Root reads the diagnostic message in its file, a determination can be made whether the Possessor is in fact the true Holder or not, and proper action can be taken.

Alternatively, the merchant's equipment can automatically connect the smartcard to the card's issuer/owner. The owner first disables the smartcard and then proceeds to interact with the smartcard's possessor to determine whether the possessor has authority to possess the smartcard. If the possessor does have that authority, the issuer/owner re-enables the smartcard.

Given the above-disclosed structure and operating system of the smartcard, it is clear the issuer/owner who installs all services found on the smartcard has knowledge of those services. That is, although the issuer/owner does not have the power to delve into files owned by the various Service Providers (even though it is the Root owner of the smartcard) the issuer/owner nevertheless knows of the specific Service Providers are resident on each of its smartcards. This knowledge can be maintained in a database owned by the issuer/owner (although each smartcard can also maintain that information about itself).

In the event a smartcard is lost or destroyed, a new smartcard can be issued to Holder with all of the Service Providers installed *ab initio*. The only items that cannot be recovered are the data files created by the various users in the old file and the Service Providers' passwords. As with initial installations, a set of temporary password files can be installed. The Service Providers can then be contacted by the issue/owner to inform them of the temporary passwords, and the Holder then contact the Service Providers to modify the passwords and to populate the necessary files in their respective directories.

Audit Trail

As indicated above, Root can maintain a log file and store therein a record of each transaction. This file can then be used to keep tract of various thresholds that the Holder, or the Service Provider's may wish to impose.

Excessive uses of a smartcard can be an indication of a fraudulent use. As indicated above, such uses can be detected through careful scrutiny of the log file and stopped.

Another use of the log file relates to perfectly valid uses. For example, a credit-providing Service Provider may wish to be informed in "real time" whenever charges over a certain limit are incurred, while allowing "batch" transmissions from the merchants (perhaps at the end of the work day) for all smaller transactions. In connection with the electronic purse of a smartcard, the Holder may instruct the smartcard to automatically contact the Holder's bank when the money value in the smartcard is below a certain limit, and transfer some additional funds into the smartcard.

Still another use of the audit trail relates to dispute resolution. If a merchant asserts that the smartcard was used to obtain some goods or services and the Holder disputes the assertion, the log file can be used to resolve the dispute.

Cooperation Between Service Providers

It is quite possible for service providers to form cooperative alliances. Such alliances can specify various activities which are carried out in the smartcards whenever the smartcard is accessed, or when the smartcard is accessed by a particular user. The number of such possibilities is limitless, and the example below is merely illustrative.

Assume, for example, that company Z employs traveling salespersons who, expectedly, need to purchase gasoline fairly regularly. Z is likely to contract with O to issue a smartcard for each of Z's salespersons (Holders) and request O to install Z as a Service Provider and G as the gasoline provider. Sometime later, A may reach an agreement with bank B as a Provider of credit for the salespersons. That service can be remotely installed into all of the smartcards belonging to the salespersons by, for example, obtaining the cooperation of G.

Specifically, Z can request G to install a request for communication with O whenever a smartcard interacts with G and found to have A as a user but not B as a user. All that G needs to do is send the appropriate command when the right kind of smartcard is logged into G.

The above disclosure speaks of "smartcards" but, actually, in connection with the invention disclosed herein, what we think of is any personal information device that is intended to primarily accompany a person wherever that person goes. Thus, the intent of the claims that follow is for the term "smartcard" to encompass all apparatus that is designed for personal use and is intended, at least in some of its functions, to carry information about a person. This clearly includes, for example, cellular phones and personal communicators. They already have the electronic means necessary to enable implementing this invention (e.g., processors), and the curent expectation is that people will carry these electronic apparatus on their persons, much like people carry conventional credit cards.

Claims

1. A multiple-application smartcard including a processor and memory in connection with which there is an issuer/owner, a holder and a service provider, comprising:

   an operating system that includes a tree-like file structure that begins with a file having characteristics that are controlled solely by said issuer/owner;

   a plurality of files, forming part of the tree-like structure and each having file characteristics that are controlled solely by said issuer/owner, which files are executable files, in the sense that, when referenced, they operate on permissible data in said memory;

   a password file that is accessible only to said issuer/owner, which contains data and which is accessed by said issuer/owner prior to said issuer/owner gaining access to said executable files;

   a password file that is accessible only to said holder, which contains data and which is accessed by said holder prior to said holder gaining access to said executable files; an

   a password file that is accessible only to said service provider, which contains data and which is accessed by said service provider prior to said service provider gaining access to said executable files.

2. A smartcard adapted for communicating with a party and including a processor comprising:

   a memory arranged into a plurality of information storage and retrieval user entity logical zones, with at least two of said logical zones including sub-zones that encompass a memory segment that contains access control data, and

   control means, for allowing access by said party to a selected one of said user logical zones, including its associated sub-zone, only when said party deliver to said smartcard information related to the access control data contained in said selected one of said user logical zones.

3. In connection with a smartcard issued by a first party, where the smartcard includes an operating system having a tree-like file structure that begins with a directory-file with attributes that are controlled solely by said party, a plurality of files, forming part of the tree-like structure and each having file attributes that are controlled solely by said first party, which files are executable files, in the sense that, when referenced, they operate on permissible data in said memory, and a password file that is accessible only said first party gaining access to said executable file, a method for installing capability for a second party to provide service to a holder of the smartcard, comprising the steps of:

   said holder assisting in establishing communication between the smartcard and said first party;

   executing a log-in protocol between the smartcard and said first party, employing the data contained in said password file;

   communicating to said first party a request for installation of a service capability on said smartcard for said second party;

   said first party establishing a user password file in said smartcard, with said user password file arranged to form part of said tree-like structure;

said first party inserting data into said user password file; and

said first party changing file attributes of said user password file to make it accessible only to said second party.

4. The method of claim 3 further comprising a step of informing said second party of the data stored in the user password file.

5. The method of claim 3 further comprising a step, executed by said first party following said step of communicating to said first party a request for installation , of confirming with said second party that the request for installation of service should be fulfilled.

6. The method of claim 3 wherein said communication is employing a telecommunication network.

7. A method for a party to communicate with a personal information device comprising the steps of:

the personal information device authenticating the party through a challenge-response sequence; and

the party authenticating the personal information device through a challenge-response sequence.

8. The method of claim 7 in which the step of the party authenticating the personal information device precedes the step of the personal information device authenticating the party.

9. The method of claim 7 further comprising a step of a holder of the personal information device being authenticated to the personal information device through a challenge-response sequence.

10. The method of claim 7 where the step of the personal information device authenticating the party is incomplete when the step of the party authenticating the personal identification device is commenced.

11. The method of claim 7 where the step of the personal information device authenticating the party comprises the steps of:

the personal information device sending a challenge that include ID information and a first data string;

the party encrypting the first data string and sending the encrypted string to the personal information device, where the key used to encrypt the first data string is based on the ID information sent by the personal information device; and

the step of the party authenticating the personal information device comprises the steps of:

the party sending a challenge that includes a second data string;

the personal information device encrypting the second data string and sending the encrypted second data string to the party, where the encryption key used for encrypting the second data string is pre-stored in the personal information device; and

the party confirming the authenticity of the personal information device by virtue of the encryption key used to encrypt the second data string; and

the party authenticating the personal information device by virtue of the encryption key used to encrypt the second data string.

12. The method of claim 11 where the first and the second data strings comprise random sequences.

13. The method of claim 11 where the first and the second encryption keys are the same.

FIG. 1

PRIOR ART

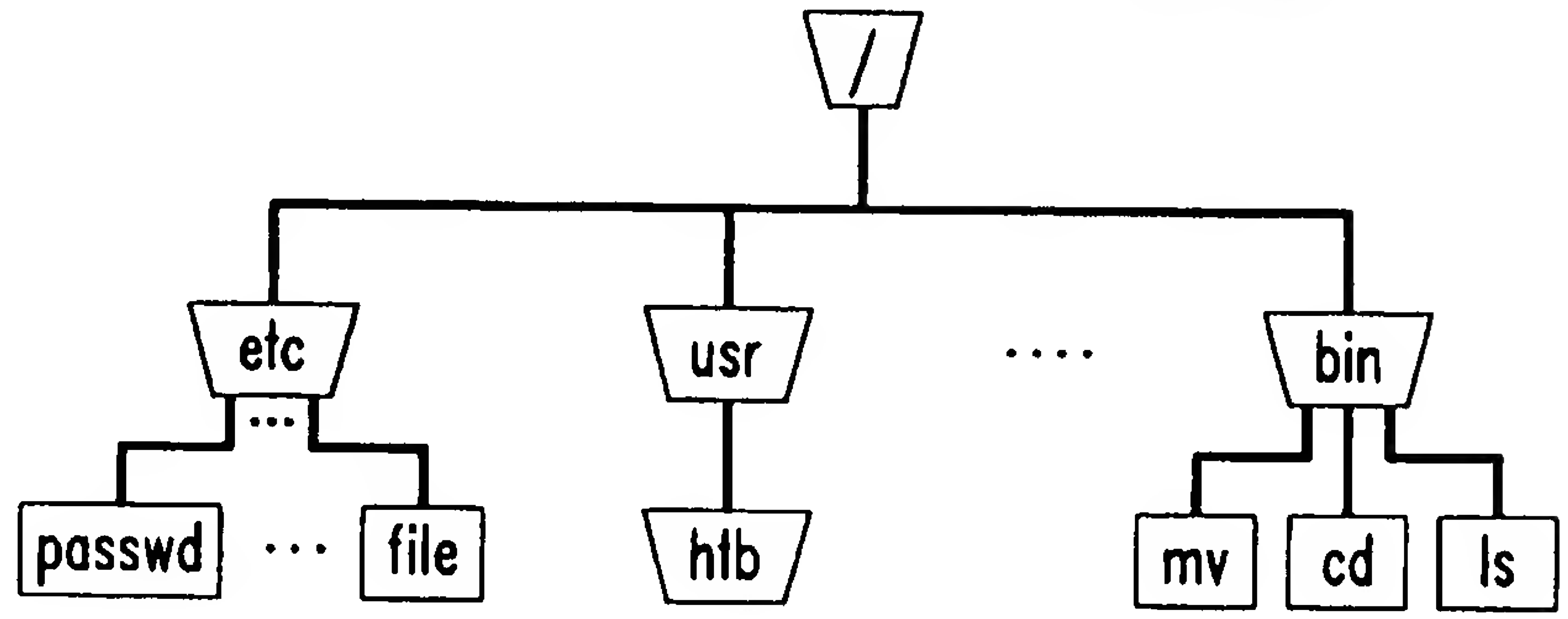


FIG. 2

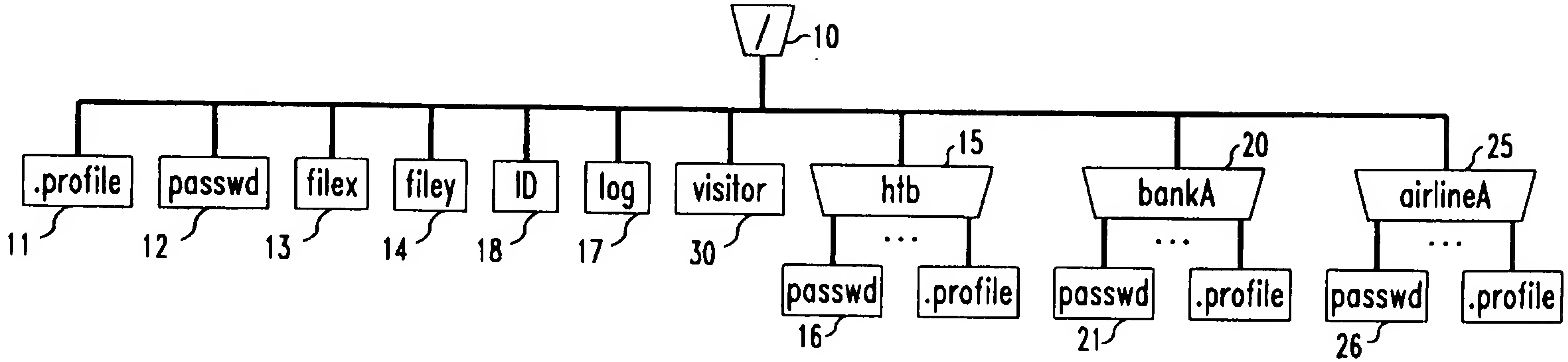


FIG. 3

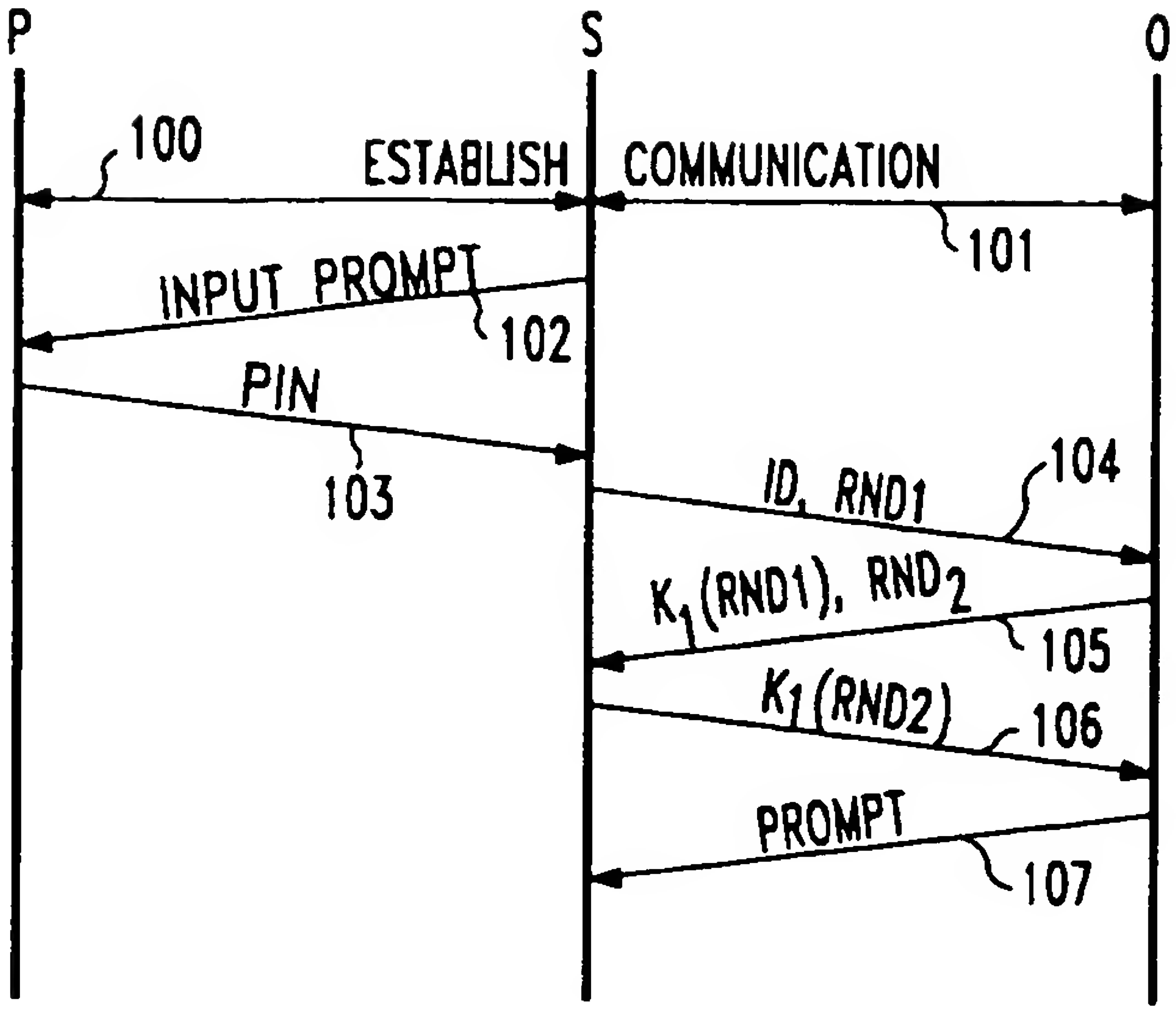

P
S
O
100
ESTABLISH
COMMUNICATION
101
INPUT PROMPT
102
PIN
103
ID, RND1
104
K1(RND1), RND2
105
K1(RND2)
106
PROMPT
107

FIG. 5

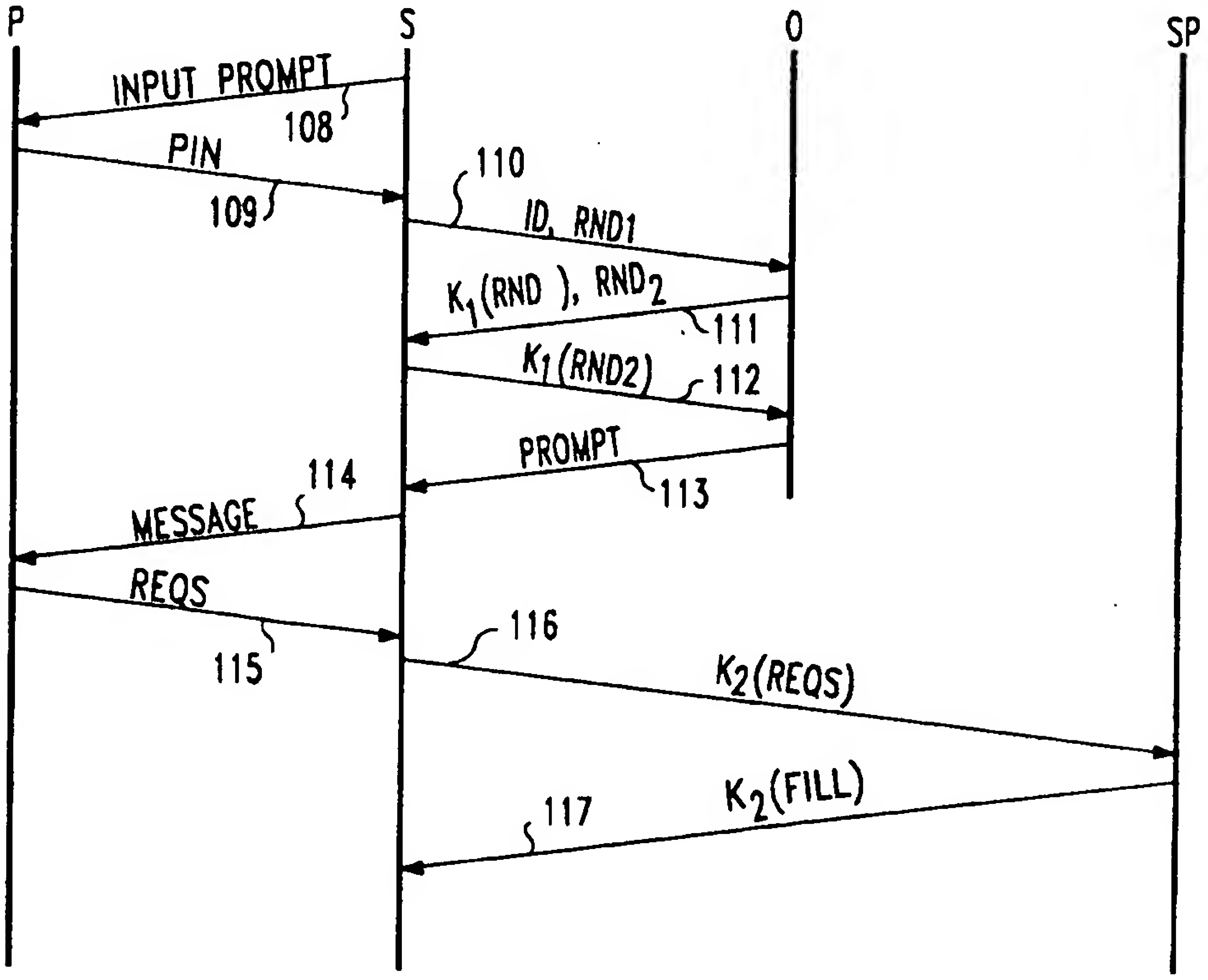

P
S
O
SP
INPUT PROMPT
108
PIN
109
110
ID, RND1
K1(RND), RND2
111
K1(RND2)
112
PROMPT
113
114
MESSAGE
REQS
115
116
K2(REQS)
117
K2(FILL)

FIG. 4

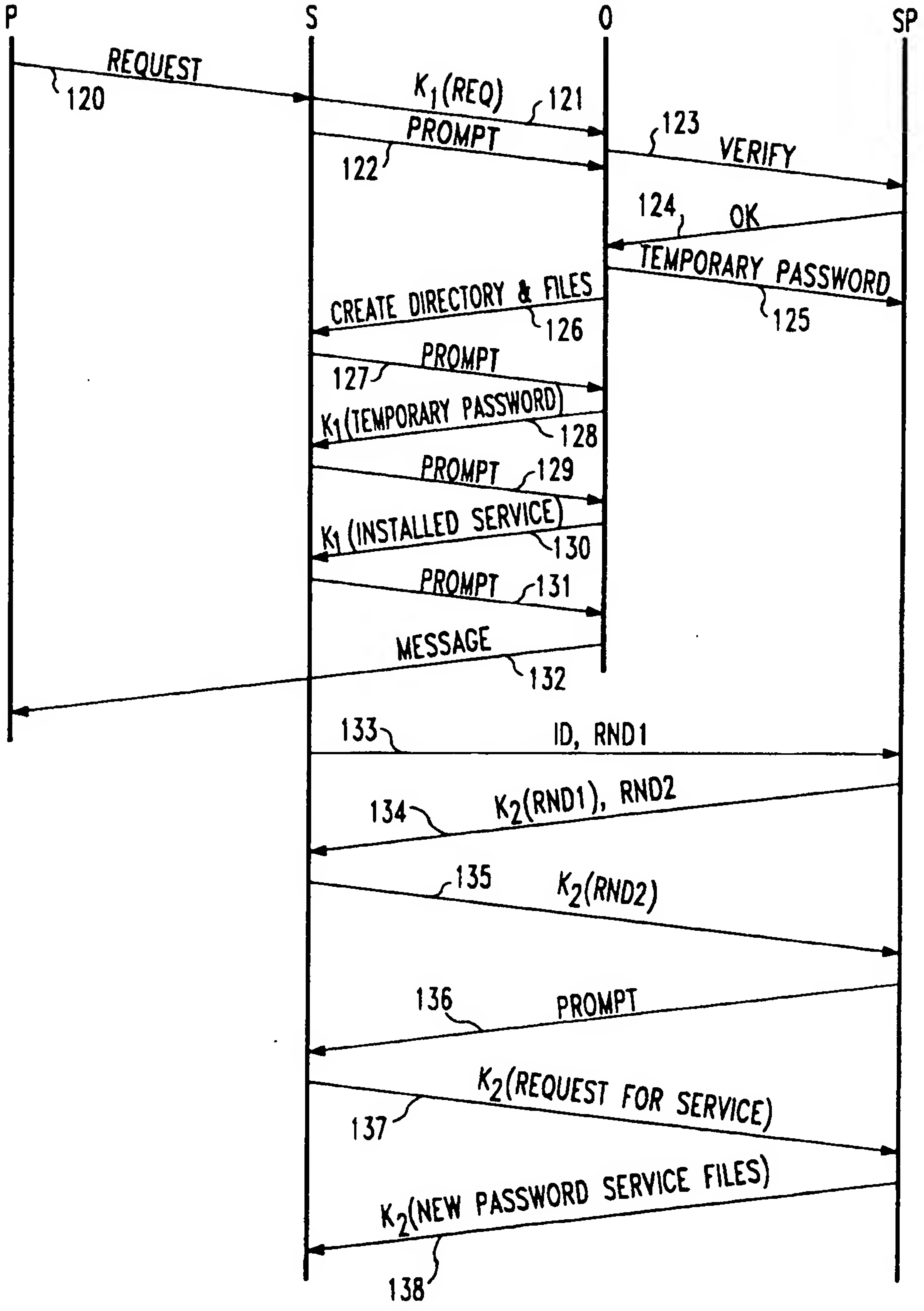

P
S
O
SP
REQUEST
120
K1(REQ)
121
PROMPT
122
123
VERIFY
124
OK
TEMPORARY PASSWORD
125
CREATE DIRECTORY & FILES
126
PROMPT
127
K1(TEMPORARY PASSWORD)
128
PROMPT
129
K1 (INSTALLED SERVICE)
130
PROMPT
131
MESSAGE
132
133
ID, RND1
134
K2(RND1), RND2
135
K2(RND2)
136
PROMPT
K2(REQUEST FOR SERVICE)
137
K2(NEW PASSWORD SERVICE FILES)
138

FIG. 6

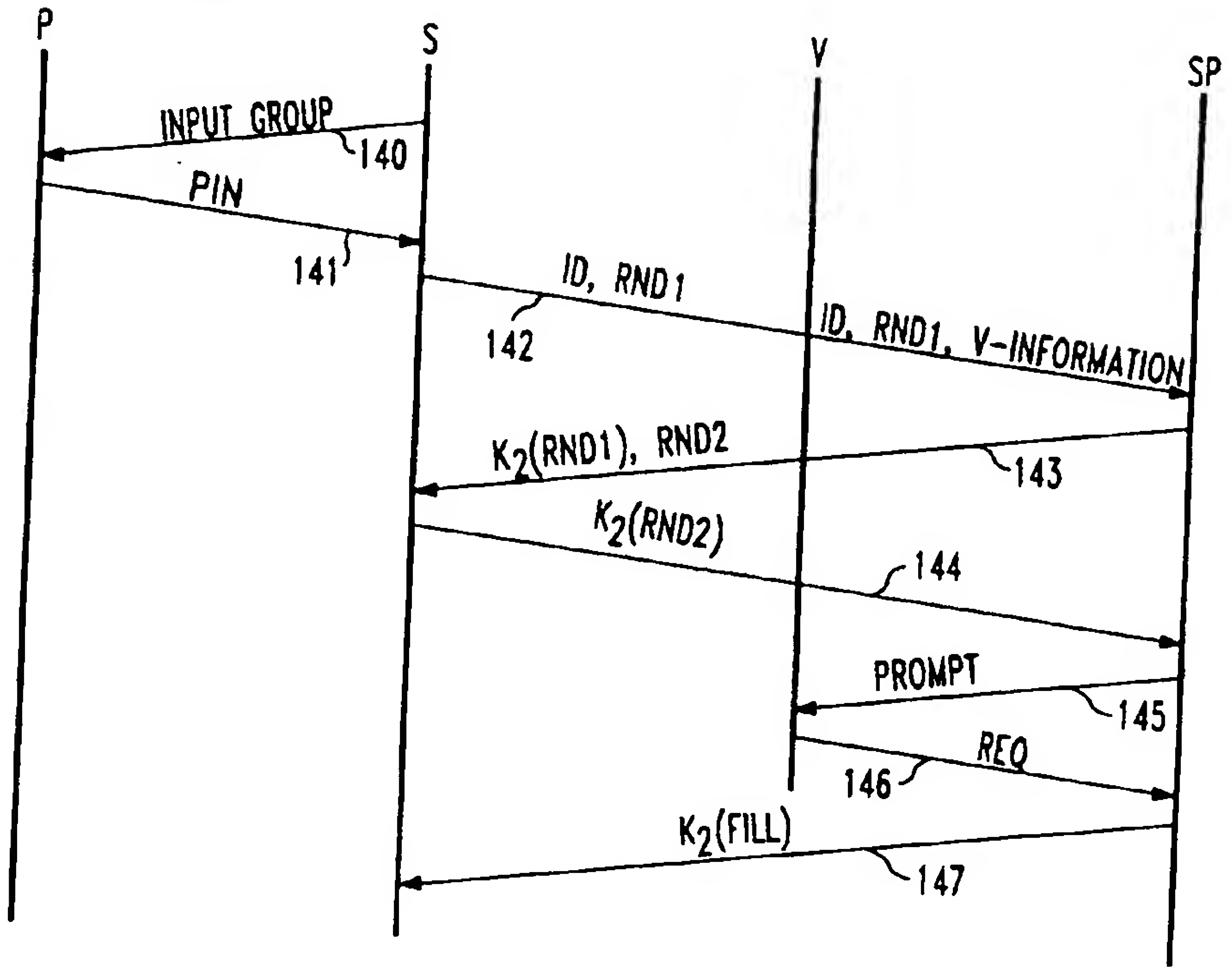

P
S
V
SP
INPUT GROUP
140
PIN
141
ID, RND1
142
ID, RND1, V-INFORMATION
K2(RND1), RND2
143
K2(RND2)
144
PROMPT
145
REQ
146
K2(FILL)
147

FIG. 7

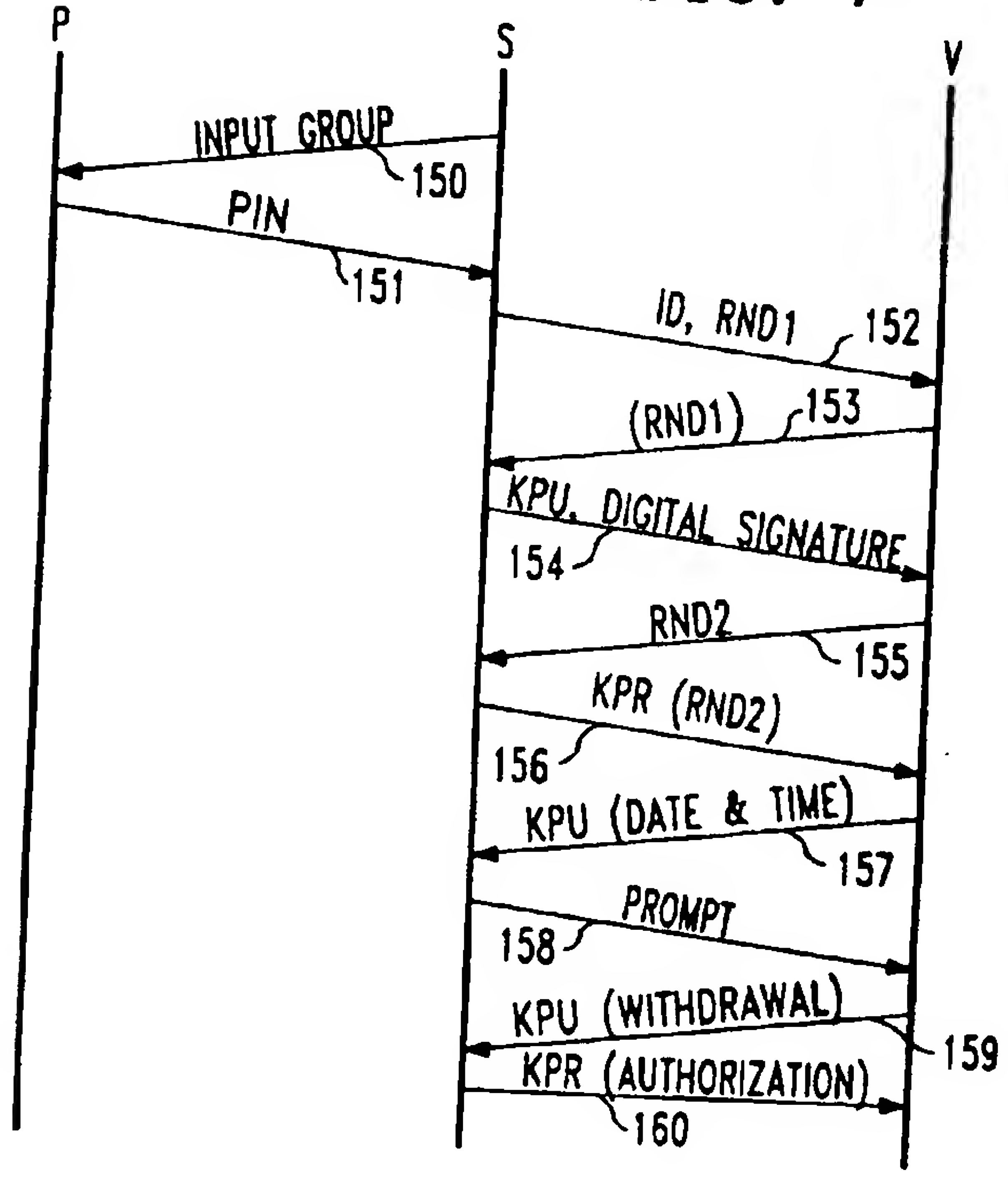

P
S
V
INPUT GROUP
150
PIN
151
ID, RND1
152
(RND1)
153
KPU, DIGITAL SIGNATURE
154
RND2
155
KPR (RND2)
156
KPU (DATE & TIME)
157
PROMPT
158
KPU (WITHDRAWAL)
159
KPR (AUTHORIZATION)
160

FIG. 8

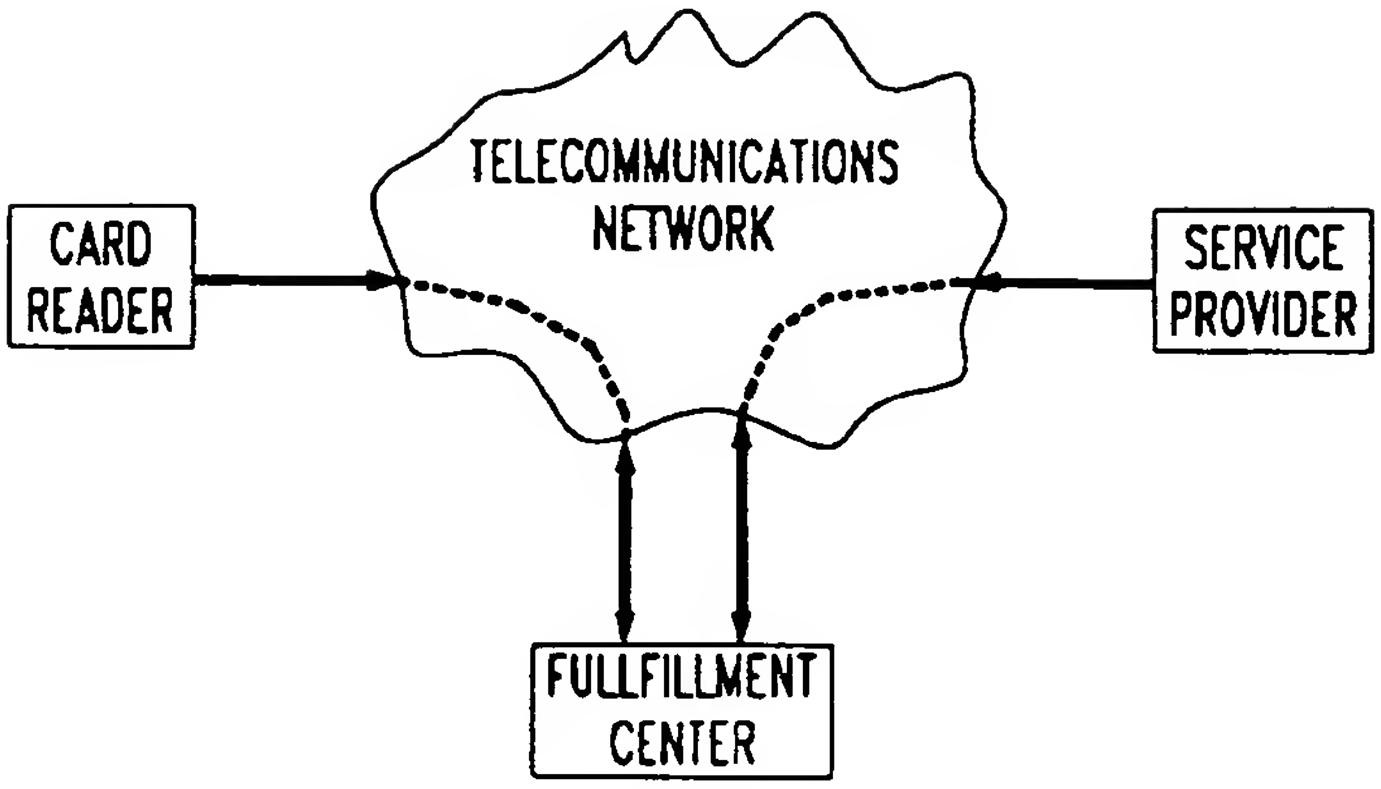
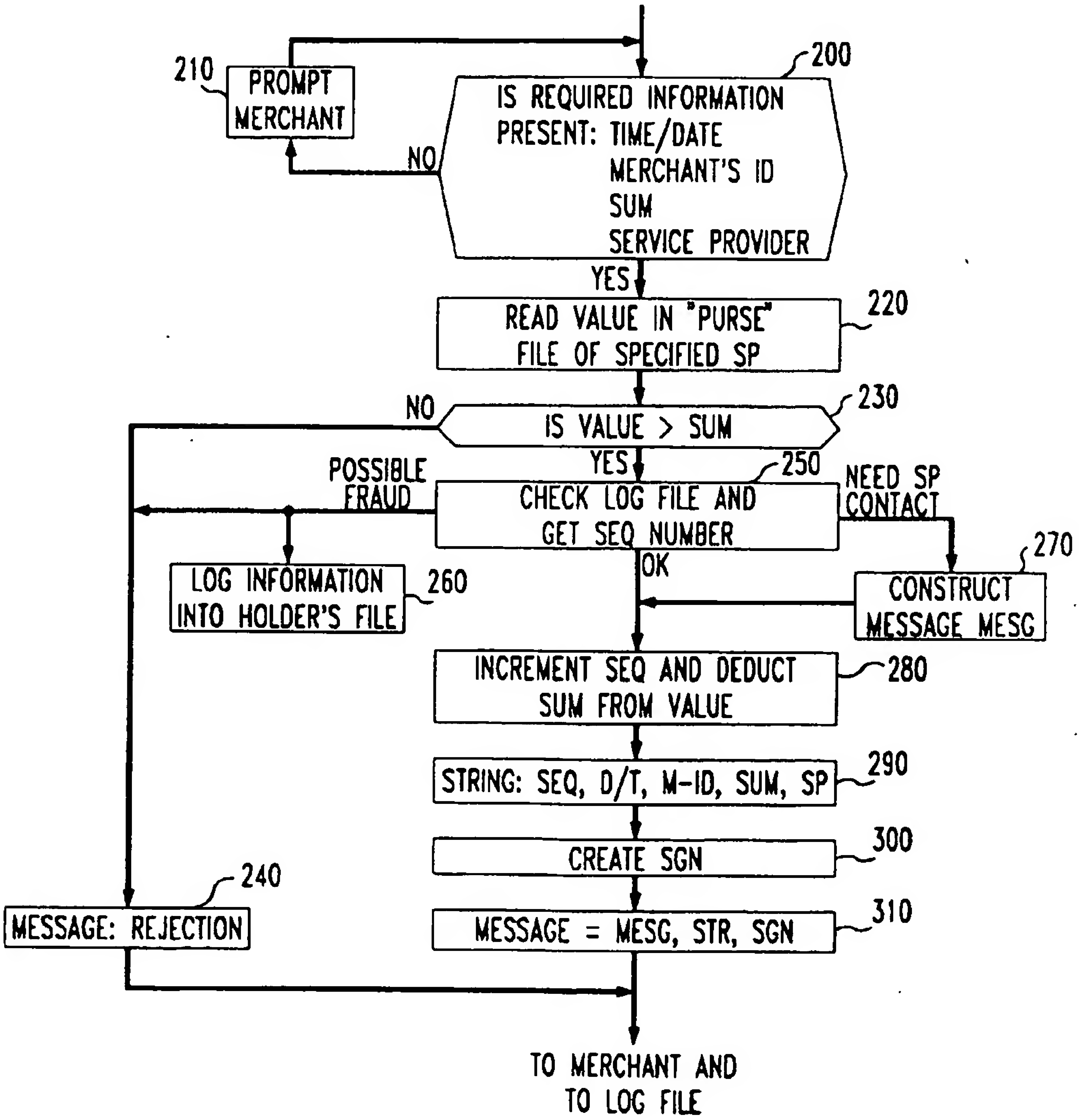

TELECOMMUNICATIONS NETWORK
CARD READER
SERVICE PROVIDER
FULLFILLMENT CENTER

FIG. 9


210
PROMPT MERCHANT
200
IS REQUIRED INFORMATION PRESENT: TIME/DATE
MERCHANT'S ID
SUM
SERVICE PROVIDER
NO
YES
READ VALUE IN "PURSE" FILE OF SPECIFIED SP
220
NO
IS VALUE > SUM
230
YES
250
POSSIBLE FRAUD
CHECK LOG FILE AND GET SEQ NUMBER
NEED SP CONTACT
OK
LOG INFORMATION INTO HOLDER'S FILE
260
270
CONSTRUCT MESSAGE MESG
INCREMENT SEQ AND DEDUCT SUM FROM VALUE
280
STRING: SEQ, D/T, M-ID, SUM, SP
290
CREATE SGN
300
240
MESSAGE: REJECTION
MESSAGE = MESG, STR, SGN
310
TO MERCHANT AND TO LOG FILE